MudOS v21c2 中文翻译文件总索引

Compile to CHM By Guan.Yun

外部函数 [efuns]

驱动程序调用的物件函数 [applies]

LPC 的概念 [concepts]

LPC 语言 [lpc]

LPC 教学

基础 LPC

中阶 LPC

目前大致上已将 MudOS v21c2 之原文文件翻译完成。不过，因为译者所学不多, 难免有遗珠之憾。未能翻译之文件清单列于附注一。

这份文件中的分类结构如下:

- applies —— 驱动程序对各个物件调用的函数
  - general —— 对一般物件调用的函数
  - interactive —— 对玩家物件调用的函数
  - master —— 对主控物件调用的函数
- concepts —— 撰写 LPC 所必备的一些概念
- lpc —— 有关于 LPC 语法的资料
  - constructs —— LPC 程序的保留字
  - preprocessor —— LPC 程序的预处理命令

        - types —— LPC 的变量、函数的类型定义
    - efuns —— 各个外部函数的使用手册
        - arrays —— 数组
        - buffers —— 缓冲区
        - calls —— 调用
        - compile —— 编译
        - ed —— ed 编辑程序
        - filesystem —— 文件系统
        - floats —— 浮点数
        - functions —— 函数
        - general —— 一般
        - interactive —— 互动物件
        - internals —— 内部
        - mappings —— 映射
        - mudlib —— 函数库
        - numbers —— 数字
        - objects —— 物件
        - parsing —— 语法分析
        - sockets —— 套接字
        - string —— 字符串
        - system —— 系统

文件中如有错误或是更好的翻译，请来函指正。如果您愿意翻译附注一中所列之文件，也请来函告知，并允许译者收录于往后的版本中。

附注一：

以下二十九份文件为译者所力有未逮之处....:Q

doc/concepts/oop
doc/concepts/socket_efuns
doc/defines 共十份文件
doc/defines 共十一份文件
doc/driver 共五份文件
doc/lpc/types/substructures

附注二：

另有 efuns 十六份文件承 Kenny 热情赞助, 详见外部函数之附注.

附注三：

以下所采用之中文译名, 文件中大多不再补注其英文原名.

admin	mud 系统管理员
administrator	admin 的原文, 但在此通常指主机管理员
Annihilator	安老大，消灭者
argument	参数
array	数组
buffer	缓冲区
character	字符
driver	驱动程序
efun	外部函数, external function
element	元素
euid	effective user id 之缩写. 译为有效使用者识别名称.
flag	标志
force	原力
function	函数或函数指针
global	全局
heartbeat	心跳
id	识别名称
information	信息或数据
interger	整数
local	本地、区域
mapping	映射
message	消息
mixed	译为「混合」, 此为 LPC 所特有的数据类型, 允许此数据为任何种类的数据类型.
null	空值
object	物件

player	玩家
pointer	指针
query	查询
return	返回 (动词)
runtime	编译时段
string	字符串
source code	源代码
system	系统
target	目标
uid	user id 之缩写. 使用者识别名称
value	值
void	原意为「无」, 视情况翻译为无返回值或无参数.
wizard	巫师
zero	零或零值

回到上一页

? ??????efuns??

?????????????? ?????ķ??????????

Spock @ FF 97.Jul.26.

? ???? ???? ???? ????

??????????eķ????ļ???

ص???hx

---

allocate(3)�� MudOS v21c2�� (5 Sep 1994)

������:

�� allocate() - ������ڟ��h������ (array).

���:

�� mixed *allocate(int size);

�� ��� *allocate(���� size);

���÷�:

��
�������ڟ��h����
size �� ��(element)�� ������.
ص���L�������
0, ���X����Ɋ���ϵ΢趨������ (ɕ��L 10000 ����) .
���� ص ij’
(initialized) ��Ï 0.

���o:

�� sizeof(3), allocate_mapping(3)

������:

�� Spock @ FF 96.Oct.12.�� (printed 3/16/95)

arrayp(3)�� MudOS v21c2�� (5 Sep 1994)

������:

�� arrayp() - ��h������ı����p�ï���� (array).

���:

�� int arrayp(mixed arg);

�� ���� arrayp(��� arg);

���-�:

�� ��� arg ��h������, ѝ��� 1.

���o�:

�� mapp(3), stringp(3), objectp(3), intp(3), bufferp(3), floatp(3), functionp(3), nullp(3), undefinedp(3), errorp(3), pointerp(3)

������:

�� Spock @ FF 96.Oct.15.�� (printed 3/16/95)

filter_array(3)������ MudOS v21c2�� (5 Sep 1994)

������:

�� filter_array() - ����h�����ے�������� ��������.

���:

```
�� mixed *filter_array( mixed *arr, string fun, object ob, mixed extra, ...
);
�� mixed *filter_array( mixed *arr, function f, mixed extra, ... );

�� ��� *filter_array( ��� *arr, ۔��� fun, ��� ob,
��� extra, ... );
�� ��� *filter_array( ��� *arr, ���� f, ��� extra, ... );
```

���-�:

```
�� filter_array()�� ����h�������,
������������ arr ������ ob ەeĺ��  fun��
����ۜ��Ľ��. ÿh������������ arr ەe  ��
(element)��  ����ú���  fun��  h��, ���ɰ���  fun
ەIJ���  (parameter). ۔۔۔۔۔۔۔۔۔  extra��
ۜ۔۔IJ���, ����ÿhەε��ú���  fun��
ﬧ۔۔۔ɵJ۔۔۔۔۔۔۔۔۔۔۔۔۔۔۔۔....h۔۔۔۔۔۔۔۔  fun۔۔
۔۔۔۔  1۔۔ۜ ۔۔, ۔۔۔۔Ïۂɾ۔۔۔۔۔  fun۔۔  ۂĹ۔۔۔
˜(۔۔۔۔۔Ҳ۔۔۔۔۔۔. (۔۔۔۔۔۔۔۔ص۔۔ڸ
ڸڸ۔۔۔۔۔  ۔۔,
۔۔۔۔۔۔IJ۔۔۔۔Á۔۔۔  f, ۔۔۔۔۔h۔۔۔  ەe۔  ob-
>fun()۔۔  ۔۔۔÷۔۔۔. ۔۔۔  arr۔۔  ۔۔۔۔h۔۔۔۔۔۔۔,
filter_array()۔۔  ۔۔۔۔  0.
```

۔۔۔o۔:

```
۔۔ sort_array(3), map_array(3)
```

map_array(3)�� MudOS v21c2�� (5 Sep 1994)

������:

�� map_array() -
����h������ h������ ⁄⁄ (element)

���:

```
�� mixed *map_array( mixed *arr, string fun, object ob, mixed extra, ...
);
�� mixed *map_array( mixed *arr, function f, mixed extra, ... );

⁄⁄ ⁄⁄⁄ *map_array( ⁄⁄⁄ *arr, ⁄⁄⁄⁄ fun, ⁄⁄⁄ ob, ⁄⁄⁄
extra, ... );
⁄⁄ ⁄⁄⁄ *map_array( ⁄⁄⁄ *arr, ⁄⁄⁄⁄⁄⁄ f, ⁄⁄⁄ extra, ...
);
```

⁄⁄⁄⁄⁄:

⁄⁄ ⁄⁄⁄⁄ ob->fun()⁄⁄ ⁄⁄ f⁄⁄ ⁄⁄⁄⁄ÿh⁄⁄ arr⁄⁄
⁄e⁄ ⁄⁄⁄⁄, ⁄⁄⁄⁄h⁄⁄⁄µ⁄⁄⁄⁄⁄. ÿh⁄⁄ arr⁄⁄
⁄e⁄ ⁄ p⁄⁄⁄⁄⁄⁄⁄⁄⁄IJ⁄⁄⁄⁄⁄⁄⁄⁄⁄⁄h⁄⁄⁄⁄⁄Í
⁄⁄⁄⁄ extra⁄⁄
⁄⁄⁄⁄ÿ⁄ε⁄⁄ú⁄⁄⁄ƒ⁄⁄⁄ɵJ⁄⁄⁄⁄ƇIJ⁄⁄⁄.
⁄⁄⁄M⁄⁄⁄⁄⁄⁄⁄⁄⁄⁄⁄⁄⁄⁄:

```
⁄⁄ foreach (index) arr[index] = ob->fun(arr[index],extra);
```

⁄⁄ ob->fun(arr[.index.], extra) ⁄⁄⁄ص⁄⁄⁄⁄滙
⁄⁄⁄⁄⁄⁄⁄⁄e⁄ ⁄⁄. ⁄⁄⁄ arr ⁄⁄⁄⁄h⁄⁄⁄⁄⁄⁄, ⁄⁄
map_array()⁄⁄ ⁄⁄⁄⁄ 0.

⁄⁄⁄⁄o⁄:

⁄⁄ filter_array(3), sort_array(3), map(3)

⁄⁄⁄⁄⁄⁄:

member_array(3)�� MudOS v21c2�� (5 Sep a994)

������:

�� member_array() -�� ��h������ (array)�� ����_��� (string) ����Ψ_������Ŀ (item) , ������ش���ĿĬ؟ن.

���:

�� int member_array(mixed item, mixed * | string arr, void | int start);

�� ���� member_array(��� item, ��� * �� ?_�� arr, ? �� ?? ?? ? start);

���-?:

?? ???? item ?????? arr?? ?e?h?γ???ħ, ?????ل???. ???'?? start ????, ??? start ???ژن??ً??????. ????Χ??? item, ? ? -1.

??????:

?? Spock @ FF 97.Apr.15.?? (printed 3/16/95)

---

pointerp(3)�� MudOS v21c2�� (5 Sep 1994)

������:

�� pointerp() -�� ж̣����ı����p̈h������ (array).

���:

�� int pointerp(mixed arg);

�� ���� pointerp(��� arg);

���-�:

�� ��� arg ��h������, �� ̈ 1.

���ö:

�� mapp(3), stringp(3), objectp(3), intp(3), bufferp(3), floatp(3), functionp(3), nullp(3), undefinedp(3), errorp(3), arrayp(3)

������:

�� Spock @ FF 97.Apr.24.�� (printed 3/16/95)

---

sort_array(3)�� MudOS v21c2�� (5 Sep 1994)

������:

�� sort_array() - ��h������������������.

���:

```
�� mixed *sort_array( mixed *arr, string fun, object ob );
�� mixed *sort_array( mixed *arr, function f );
�� mixed *sort_array( mixed *arr, int direction );

�� ��� *sort_array( ��� *arr, ����_ fun, ��� ob );
�� ��� *sort_array( ��� *arr, ������_ f );
�� ��� *sort_array( ��� *arr, ����_ direction );
```

���-�:

```
�� ��h���  ���h����_ arr��
����r  ص������,
��������e  ��_������ ob->fun()
��_���Ĺ�������_��_ (ascending) ����.
ÿ�ε��  ob->fun() ʰ, ��п_��_. fun()��
Ö_____________Ĺ�ϵ____ -1, 0, 1 (��C,
____, ��).

��
ĵ____  ___h��h��,
`'ú____ f��  _____
fun.

��
_______'  ___ŏ_____r
(built-in sort routine). direction Ï 1��  ____ 0 ,��
_____________, ___________, ��  direction Ï -1
_______½_ ,��
(descending)
```

unique_array(3)�� MudOS v21c2�� (5 Sep 1994)

������:

	�� unique_array() - ��h������������������.

���:

	�� mixed unique_array(object *obarr, string separator, void | mixed skip
	);
	�� mixed unique_array(mixed *arr, function f, void | mixed skip);

	�� ��� unique_array(��� *obarr, ̣��� separator,
	� ��� �� ��� skip);
	�� ��� unique_array(��� *arr, ̣̣̣̣̣̣ f, ̣ ̣ ̣

```
```

```
```

```
```

? ???? ???? ??????

??????????ek????ļ???

ص???hx

---

allocate_buffer(3) MudOS v21c2�� (5 Sep 1994)

������:

�� allocate_buffer() - ������ڟ��h�������

���:

�� buffer allocate_buffer(int size);

�� ******* allocate_buffer(****** size);

���÷:

�� *****h***** size �� ** (elements) ***********.
*****L********* 0, ***X***η***ϵ***Ͳ*趨
*********(*****10000 *****) .
********** ***ij*’ (initialized) **ï 0.

***o*:

**** bufferp(3), read_buffer(3), write_buffer(3), sizeof(3), to_int(3)

****:

**** Truilkan

******:

**** Spock @ FF 96.Oct.12.**** (printed 3/16/95)

---

bufferp(3)�� MudOS v21c2�� (5 Sep 1994)

������:

�� bufferp() -
��h�������ı����p�ïh������ (buffer).

���:

�� int bufferp(mixed arg);

�� ���� bufferp(��� arg);

���-�:

�� ��� arg ��h�������и��� 1,
���Ди��� 0.

���o�:

�� mapp(3), stringp(3), pointerp(3), objectp(3), intp(3), floatp(3), functionp(3), nullp(3), undefinedp(3), errorp(3)

������:

�� Spock @ FF���� 96.Oct.16.���� (printed 3/16/95)

---

crc32(3)�? MudOS v21c2�? (5 Sep 1994)

??????:

?? crc32() - ????h??????????????????????????????

(cycle redundency code)

???:

?? int crc32(buffer | string x);

?? ???? crc32(??????? | ???? x);

???-?:

?? ????????????????????????????? x ?? CRC-32 ??.

??????:

?? Spock @ FF 97.Jan.27.?? (printed 3/16/95)

?????hx

read_buffer(3)�� MudOS v21c2�� (5 Sep 1994)

������:

�� read_buffer() - ��Ի������������ (buffer)
������и����ļ��������,
��������_������ (string)
��� ������h��� ������.

���:

�� string | buffer read_buffer(string | buffer src, int start,�� int len);

�� ��_��� �� ������ read_buffer(��_��� ��
������ src, ���� start, ���� len);

���-�:

�� �� src�� ��h����_��� (��ļ�����) , ��ɔ��
start��_ ���ĵ̨ʲ (byte) ��', ��ą len��
.�����ą��ļ�����. ��ą����
���Ի������������и���. ������_��
start �� len , ����ą�����ļ�������.

�� ��� src�� ��h��������, ��' start��
.���ĵ̨ʲλ ��'��ą len ���ą�����.
Ȼ������_������������и���.

�� ��ע��,
����Ղ�h���ļ�������������������,
      ������ʰ����ۯ��ļ (runtime config file)�� ��ж����
maximum byte
transfer
(����ڳ�����) ����.

���o�:

�� write_buffer(3), allocate_buffer(3), bufferp(3), read_bytes(3)

write_bytes(3)

******:

** Truilkan

******:

** Spock @ FF 97.May.24.** (printed 3/16/95)

****hx

---

write_buffer(3)�� MudOS v21c2�� (5 Sep 1994)

������:

�� write_buffer() - ��h������������ (buffer)
д��h����ļ,
���ij�������������Ъ-����h�������������.

���:

�� int write_buffer(string | buffer dest, int start, mixed source);

�� ����� write_buffer(ę-���� �� ������ dest,
����� start, ��� source);

���÷-�:

�� ��� dest ��h����ļ, �� source
�����ïh������ (���� network-byte-order��
д����ļ) ��h������������������h����-���,
������ļ dest ĵ start�� ę̈́'д�� source ������.

�� ��� dest ��h������������, �� source ��ÿ
start�� ę̈́'д ����. ��� source ������,
������ network-byte-order�� ęк'д��.

���o:

�� read_buffer(3) allocate_buffer(3)

������:

�� Truilkan

������:

�� Spock @ FF 97.jul.25.�� (printed 3/16/95)

? ???? ???? ???? ????

??????????ęķ????ļ???

ص???hx

---

call_other(3)�� MudOS v21c2�� (5 Sep 1994)

������:

�� call_other() - ��������h�������eĺ���.

���:

�� unknown call_other(object ob | object *obs, string func | mixed *args, ...);

�� δ call_other(��ob ��*obs, ��func ��*args, ...);

���÷:

�ÿ���w��IJ��(argument)
������h������eĺ���.
����Ç�ÿ����õĺ�������,
����'ô�h����ń��ü鶫, func ������ ob
õ����õĺ�������, ������������к...
ηηIJ���, ��л func ��, ���func
ĵ̂h������Ĵ�������. ��call_other() ��,
�����h�������h�������,
call_other()�ɔ��h�������,
������Ẋ����h������.
����Ĵ�������h������,
������eĵ̂h��(element)
������h������(��Ç���õĺ���),
������ ɔ ����Ç����'����IJ���.

�ж ������и'�call_other() ���õķ���:

object ob ��object *obs -> func(...);
��

��ʾ:

```
���� call_other( ob, "query", "name" );
�� ����д��
���� ob->query("name");

�� ��� call_other() ĵĴ��IJ�����h�������,
X,���  ������д��:
���� call_other( ob, ({ "query", "name" }));

�� ������h��'������ķ���:
���� users()->quit();
```

������:

�� Spock @ FF 96.Oct.16.�� (printed 3/16/95)

ص���hx

---

call_out(3)�� MudOS v21c2�� (5 Sep 1994)

������:

�� call_out() - ř ���h������eĺ���.

���:

�� void call_out(string | function fun, int delay, mixed arg);

�� ̈ ̈ ̈
call_out(̈́̈́̈́ ̈ ̈́̈́̈́ fun, ̈́̈́̈́ delay, ̈́̈́̈́
arg);

���÷�:

�� ̈́̈́ this_object() ̈́̈, ̈́ř̈ delay ̈́̈́̈́ú̈́̈́ fun.
̈́̈ arg ̈́̈́̈́̈́̈́̈́̈́̈́̈́̈́ fun ̈́̈.

̈́̈ ̈́̈ʊ̈́̈, ̈́̈ŕ̈́̈́̈́ fun ̈́ë́̈́ write() ̈́̈ say(), ̈́̈Ï
this_player() ̈́̈́̈́ ̈́ 0. ̈́̈́̈́̈́̈́̈́ tell_object() ̈́̈́̈́.

̈́̈ ̈́̈́̈́̈ driver ̈́̈ options.h ̈́ӝ́̈́
THIS_PLAYER_IN_CALL_OUT, ̈́J̈́̈́ƕʊ̈́̈́̈́̈́̈́̈́̈́̈́̈́.

���ö́:

̈́̈ remove_call_out(3), call_out_info(3)

������:

̈́̈ Spock @ FF 96.Oct.16.̈́̈ (printed 3/16/95)

catch(3)�� MudOS v21c2�� (5 Sep 1994)

������:

	�� catch() - ��ϗ�������������ĵ���.

���:

	�� mixed catch(mixed expr);

	�� ��� catch(��� expr);

���-�:

	�� ���� expr '��. ���û�г���и��� 0�� ,
	����б�����, �� ̀h����h���ˍ��*
	���ˍ���.

	�� ���� throw()��
	Ϗ�������������Þ����кη���. catch()
	'ϵ���c�������������Í�������,
	���DZ����ϵ (compiler) ��h��ˍ��.

	�� catch() ��e��ķ��єт��ϑ, ϵ���� ϵ����. ϵ catch()
	ϵ��ۍ���ʰ ϵ ϵєт�ű (consistency)ϵ ϵĵطϵ.

���o�:

	ϵ error(3), throw(3), error_handler(4)

������:

	ϵ Spock @ FF 96.Oct.24.ϵ (printed 3/16/95)

������:

�� origin() - ж╨Ьй╒Н╞╬╬╬╬╬╬╬α╬╬╬╬ш╒.

���:

```
�� int origin( void );
```

```
�� ╬╬╬╬ origin( ╒   ╬╬╬ );
```

╬╬╬╓╒:

```
�� ╬╬╬╬h╬╬╬╬╬╬,
ЁЁЬй╒Н╞╬╬╬╬╬╬╬α╬╬╬╬ш╒.
```

```
ЁЁ ╬╬╬╬╬╬╬╬╬╬╬╬╬╬╬╬╬╬╬╬:
```

```
ЁЁ 1 ╬╬╬╬╬╬╬╬ (driver) (heart_beats, ╬╬╬╬)
```

```
ЁЁ 2 ╒╬╬╬╬╬╬╬й╬╬╬ (local function call)
```

```
ЁЁ 4 call_other()
```

```
ЁЁ 8 simul_efun ╬╬╬, ╬╬╬╬ simul_efun ╬╬╬╬
```

```
ЁЁ 16 call_out()
```

```
ЁЁ 32 ╬╬╬  ╬╬╬╬ (efun) ╬╬╬╬ (sort_array, ╬╬╬╬)
```

╬╬╬o╒:

```
ЁЁ previous_object(3), /include/origin.h
```

������:

```
ЁЁ Spock @ FF 97.Apr.24.ЁЁ (printed 3/16/95)
```

previous_object(3) MudOS v21c2�� (5 Sep 1994)

������:

�� previous_object() -
���ص���Ľǰ�������������������б(3).

���:

�� mixed previous_object(int | void);

�� ��� previous_object(���� �� ⊕ ��);

���÷:

�� ����h������Ľǰ������ő�������������.
��ʊ��, ę������ (local function)�� ĵ��ò������
previous_object()�� ⊕趨ĪĽǰ�����, ����ı
previous_object()�� ���ص����.
ę����h�������Ī����,
ęи������ǰٟ̊������õ����. previous_object(0) ��
previous_object(), previous_object(1) ⊕ previous object ⊕
previous_object(), ę֟ę.⊕ previous_object(-1)
⊕ ⊕⊕⊕⊕e⊕ previous object ⊕⊕⊕⊕.

���o:

�� call_other(3), call_out(3), origin(3)

������:

⊕ Spock @ FF 97.Apr.24.⊕ (printed 3/16/95)

shadowp(3)�� MudOS v21c2�� (5 Sep 1994)

������:

�� shadowp() - ж̣ррїИЎ (shadow).

���:

```
�� object shadowp( object ob );

�� ��� shadowp( ��� ob );
```

���-�:

�� ����ИЎ ob ����, ��� ob ����к������ИЎ, �� 0.

���o:

�� shadow(3), objectp(3), query_shadowing(3), valid_shadow(4)

������:

�� Spock @ FF 97.Jul.25.�� (printed 3/16/95)

����ы:

---

remove_call_out(3) MudOS v21c2�� (5 Sep 1994)

������:

�� remove_call_out() - ą��h����ў��e�� ���� (call_out).

���:

�� int remove_call_out(string fun);

�� ���� remove_call_out(ę���� fun);

���ł�:

�� ą��Ľǰ�������ў ��õ�� ���ú��� fun.
������������ ���û�ƶ¶����ў�ħ��.
������¨�� fun�� ĜĐ�û���ў �� ����,
�и��� -1.

���o�:

�� call_out(3) call_out_info(3)

������:

�� Spock @ FF 97.May.27.�� (printed 3/16/95)

---

shadow(3)�� MudOS v21c2�� (5 Sep 1994)

������:

�� shadow() - ��ÏhЩ�����h������������������ИЎ (shadow).

���:

�� object shadow(object ob, int flag);

�� ��� shadow(��� ob, ���� flag);

���-�:

�� ��� flag �� 1, ���Dz.�� flag ,
��Lj������������Ï ob�� ��ИЎ. ��� flag �� 0,
���p��� 0, ���p������������Ï ob ИЎ�����.

�� valid_shadow()�� ����������������������.
��� valid_shadow() ���� 1, L�������� ИЎ,
shadow()�� ������ 0�� ������ ob.
��
�� ������ a�� ��Ï��� b�� ��ИЎ,
�����ж b�� �� call_other(func)��
������µ��� (redirect) �� a. ������ a
û�ж��� func ����, �� call_other(func) ���� b��
���� func (����û�� a ��ÏИЎ), �� a�� ������
call_other() ���� b�� ��Í���. b �� ��� call_other()
�����L. ���� b ��ڑ��Í������ý����� b
.L�����.

�� ������������������ą��ИЎ.
  ���ÏИЎ�����, ���Ġ ИЎ�����.
  �� ĕ��,
��Ï��ИЎ�����Ҳ���Ω�����.

hx

shadowp(3)�� MudOS v21c2�� (5 Sep 1994)

������:

���:

	�� object shadowp(object ob);

	�� ��� shadowp(��� ob);

���-�:

	�� ����ИЎ ob ����, ��� ob
	�����к������ИЎ, �� 0.

���o:

	�� shadow(3), objectp(3), query_shadowing(3), valid_shadow(4)

������:

	�� Spock @ FF 97.Jul.25.�� (printed 3/16/95)

---

this_object(3)�� MudOS v21c2�� (5 Sep 1994)

������:

�� this_object() - ���ص��ô'������������������ (object pointer).

���:

�� object this_object(void);

�� ��� this_object(¦ ��¦ ��¦);

���÷¦:

�� ������������������¦.
��������h�����¦h��h��Ղ-�����¦í���
(��������¦ (inherite) ����í���), ����
this_object()�¦ ¦Ł¦.

���o¦:

�¦ this_player(3), previous_object(3), origin(3)

������:

�¦ Spock @ FF 97.Jul.25.¦ (printed 3/16/95)

throw(3)�� MudOS v21c2�� (5 Sep 1994)

������:

�� throw() - ó��h������������������.

���:

�� void throw(mixed);

�� �� ���_throw(���);

���÷�:

�� throw()��
������������ó��h������������������. ��
throw()�� �� catch()�� ����h��'��ﾡ,
ó�������� ������Ʒ������ﾡ�δ���ﾡ
(runtime error) ��'������Ш. ��'�� throw() ﾡ,
Öh��'�� catch() ����.
�� ����Ï���¸��÷�:

```
�� string err;
�� int rc;

�� err = catch(rc = ob->move(dest));
�� if (err) {
���� throw("move.c: ob->move(dest): " + err + "\n");
���� return;
�� }
```

���o�:

�� catch(3) error(3) error_handler(4)

������:

�� Spock @ FF 97.Jul.25.�� (printed 3/16/95)

? ???? ???? ????

??????????ęķ????į???

generate_source.3

ص???hx

generate_source(3) MudOS v21c2�� (5 Sep 1994)

������:

generate_source() - ������ţ�������������ĵ C ʮ����

���:

void generate_source(string file, void | string out_file);

generate_source(���ţ file, ��ţ out_file);

���÷:

generate_source() ���ʹ LPC -> C ��ı������, ������ţ������ ’��. ���û��ţ�������ĵ¦, ������ĽL’������ SAVE_BINARIES�ʹ ʹL¼.

���o:

valid_asm(4), valid_compile_to_c(4)

������:

Spock @ FF 97.Feb.13.ʹ (printed 3/16/95)

---

ed

---

ekl

---

hx

---

ed_cmd(3)�� MudOS v21c2�� (5 Sep 1994)

������:

�� ed_cmd() -�� ��h������e ed ���� (session) ��h������.

���:

�� string ed_cmd(string cmd)

�� ��ed_cmd(��cmd)

���-�:

�� ������ __OLD_ED__ ���趨�� (define) ħ��Ч.

�� ���cmd �� ��h������e ed ���(session) , ������к�L'��.

������:

�� Spock @ FF 97.Feb.12.� (printed 3/16/95)

---

ed_start(3)�� MudOS v21c2�� (5 Sep 1994)

������:

�� ed_start() - ��'��h�� ed ����.

���:

�� string ed_start(string file | void, int restricted | void)

�� ̲���� ed_start(̲���� file �� ̲ ̲ ���, ����
restricted �� ̲ ̲ ���)

���-̲:

�� ̲̲̲̲̲̲̲̲ __OLD_ED__ ���趨�� (define)
ħ��Ч.

�� ̲̲̲̲̲̲̲̲'̲̲̲گ̲̲̲̲̲̲̲̲ ̲̲̲̲,
̲̲̲̲̲̲̲̲̲̲Ç̲̲̲Ш ̲̲̲ļ file.��
̲̲̲̲̲ ̲̲̲̲̲̲ ed ̲ı ̲̲̲. ̲̲ ed
̲̲̲̲̲̲̲̲̲̲ħ, ̲̲̲̲'̲̲ ed_cmd() ̲̲ ed
̲̲̲̲̲̲̲.

̲̲ ̲̲̲ restricted ̲̲Ї 1̲̲ ,
̲̲̲и̲̲̲̲ļ̲̲̲̲̲̲̲ ̲ ̲̲ї̲̲Ч.

̲̲ h̲̲̲̲̲̲hħ̲̲̲̲̲̲̲̲̲̲h̲̲ ed ̲̲̲̲.

̲̲̲̲̲̲:

̲̲ Spock @ FF 97.Feb.12.̲̲ (printed 3/16/95)

---

query_ed_mode(3) MudOS v21c2�� (5 Sep 1994)

������:

�� query_ed_mode() - ��ğĿǰ ���� (ed) ̣e̯.

���:

�� int query_ed_mode()

�� ���� query_ed_mode()

���÷:

�� Ψ��Ŀǰ̣б ���� (ed) ̯. query_ed_mode() ̣ķ̣̈:
��
�� 0 - ̣̣̣

? ???? ???? ?ļ?ϵT

??????????ęķ????ļ???

ص???hx

---

cp(3)���� MudOS v21c2�� (5 Sep 1994)

������:

���:

	�� int cp(string src, string dst);

	�� ���� cp(ʟ��ɴ src, ʟ��ɴ dst);

���ֽ:

	�� ������ļ src ĭ�ļ dst.

������:

	�� ������Yı��и��ɴ 1, ʃ��� ط.

���o:

	�� rm(3), rmdir(3), rename(3), link(3)

������:

	�� Spock @ FF 97.Jan.27.�� (printed 3/16/95)

---

file_size(3)�� MudOS v21c2�� (5 Sep 1994)

������:

�� file_size() - ģ����ļ��Ĵ�C.

���:

�� int file_size(string file);

�� ���� file_size(̄��� file);

���÷:

�� file_size() ������ļ file ��λ ��. ���� -1 ����
file ������, ��������Ę�� ���д���ļ.
���� -2 ����� file�� ��h��L¼.

���o:

�� stat(3) get_dir(3)

������:

�� Spock @ FF 97.Feb.12.� (printed 3/16/95)

get_dir(3)�� MudOS v21c2�� (5 Sep 1994)

������:

�� get_dir() - ������й�h����ļ�єтĽ¼ (filesystem directory)�� ����Щ.

���:

```
�� mixed *get_dir( string dir, int default: 0 );

�� ��� *get_dir( ̲��� dir, ���� Ц��̲: 0 );
```

���-�:

```
�� ��� dir�� ��h����ļ������ (����′�� *�� ����? ?�� ̲̲̲̲̲̲̲̲̲̲̲̲ (wildcards) ),
�  ̲h̲̲̲̲̲̲̲̲̲̲̲̲
```

����o�:

•• file_size(3), stat(3), time(3)

������:

•• Spock @ FF 97.Feb.14.•• (printed 3/16/95)

ص���hx

---

link(3)���� MudOS v21c2�� (5 Sep 1994)

������:

�� link() - ��h���ļ�����h���ļ�����

���:

�� void link(string original, string reference);

�� �� ��ɽ link(ɽ���
original, ɽ���ɽ reference);

���÷:

�� ���ļ reference�� ���þ�ļ original . link()��
��'������ (master object)�� ���ɽ valid_link(original, reference). ��ɽ valid_link() ���ɽ 0, ����ɛ link() ɻ��.
��ɽ valid_link() ���ɽ 1, ��Ç�ɽ rename()
�����IJ��ɽ (argument) ����, ɽж��þ�ı. ��ɽ rename(), valid_link()�ɽ ������ 1, ɽ link() ɽı.

�ɽ ʊ��: link() ������h��ó���ɽ (hard link) , ���þ������ (symbolic link).

���o:

�ɽ rm(3), rmdir(3), rename(3), mkdir(3), cp(3)

������:

�ɽ Spock @ FF 97.Feb.18.ɽ (printed 3/16/95)

mkdir(3)�� MudOS v21c2�� (5 Sep 1994)

������:

���:

	�� int mkdir(string directory);

	�� ���� mkdir(̲��� directory);

���-�:

	�� ����̲����L¼. ̲ı������ 1, ʃ��� ̲ 0.

���o�:

	�� rm(3), rmdir(3), link(3)

������:

	�� Spock @ FF 97.Apr.15.�� (printed 3/16/95)

read_bytes(3)�� MudOS v21c2�� (5 Sep 1994)

������:

�� read_bytes() - ��h���ļ�����h���_������ďh������������ ������ (byte).

���:

�� string read_bytes(string path, int start, int length);

�� ��_���� read_bytes(��_���� path, ����_ start, ����_ length);

���÷:

�� ������������ď path ������ļ������, ��_ start��_ ���ĵ̃ۏ’��ď��_ length ̣_�����ɳ̣_�����. ���ص��ị̈_������������. ʊ��, start + length�_ ĵ̂C���Ɓ_����Ɋ_����ļ���ĩ��, ____ read_bytes()_ ��ḍ̌Ķ_____ʧ��. ̣_������_ĵ̇_̣_̣_̣

read_file(3)�� MudOS v21c2�� (5 Sep 1994)

������:

�� read_file() - ��h���į������ ���h���_�������������.

���:

�� string read_file(string file, int start_line, int number_of_lines);

�� ����_ read_file(����_ file, ���_ start_line, ���_ number_of_lines);

���÷�:

�� ��h���į��ж�� ������_h���_�����. ĵ_j_____ ,_____w��. ____ file ĲJ___, ,_____į____ ____h___ _____. start_line�� ��_____Ç___į��ĵ_ų_п_’_ḑ_ ,_____ _____ 0.

���o�:

�� write_file(3), read_buffer(3)

������:

�� Spock @ FF 97.May.24.�� (printed 3/16/95)

rename(3)�� MudOS v21c2�� (5 Sep 1994)

������:

���:

	�� int rename(string src, string dst);

	�� ���� rename(ʻ��ʻ src, ʻ��ʻ dst);

���÷:

	�� ʻ㄃ʻ src ���ļ������ dst.

������:

	�� ������ı��и��ʻ 0.�ʻ ʃʻ ط.

���o:

	�ʻ rm(3), rmdir(3), cp(3), link(3)

������:

	ʻ Spock @ FF 97.May.27.ʻ (printed 3/16/95)

---

rm(3)���� MudOS v21c2�� (5 Sep 1994)

������:

	�� rm() - r��h���ļ.

���:

	�� int rm(string file);

	�� ���� rm(��_�� file);

���÷:

	�� r�� file ����ļ. ���ʧ и��� 0, ��r�� ! 1.

���o:

	�� mkdir(3), rmdir(3), link(3)

������:

	�� Spock @ FF 97.Jun.2.�� (printed 3/16/95)

---

rmdir(3)�� MudOS v21c2�� (5 Sep 1994)

������:

	�� rmdir() - r��h��L¼.

���:

	�� int rmdir(string dir);

	�� ���� rmdir(��_��� dir);

���-�:

	�� r�� dir ���L¼.

������:

	�� ��ı��и��ط���_(nonzero), ŧ��� ¢ 0.

���o:

	�� rm(3), mkdir(3), link(3)

������:

	�� Spock @ FF 97.Jun.2.�� (printed 3/16/95)

---

stat(3)���� MudOS v21c2�� (5 Sep 1994)

������:

�� stat() - ������h���ļ���h��Ł¼��ص���Ш.

���:

�� mixed stat(string str, int default: 0);

�� ��� stat(˙��� str, ���� Ц��: 0);

���÷:

�� ��� str ��h�����ˇ�ļ������ (����Ł¼),
�� stat() ���h������. ������鏾
ий��ӷ��ļ�����Ш. ����к‘‘����:

�� ({ file_size, last_time_file_touched, time_object_loaded })

�� ({ ˙ļ������, ���h˙ек˙˙˙˙˙˙˙,
time_object_loaded })

�� ��� str˙˙ ��h˙˙Ł¼, ���ģJ˙˙˙˙˙˙˙ї -1 ,
�� stat() ˙˙ get_dir(3) ˙˙˙˙˙˙˙˙˙.

���o:

�� get_dir(3) file_size(3)

������:

�� Spock @ FF 97.Jul.23.˙˙ (printed 3/16/95)

tail(3)���� MudOS v21c2�� (5 Sep 1994)

������:

�� tail() - ��'h���ļ�������h����.

���:

�� int tail(string path);

�� ���� tail(ŗ��� path);

���÷:


```
�� ���  ����  (efun) ��'ļ  path ������h����.
ıı  1, ���ı�����  0.
(������ļ���dĘ   ���)
```


���o:

�� read_file(3) read_buffer(3) file_size(3)

������:

�� Spock @ FF 97.Jul.24.�� (printed 3/16/95)

---

write_bytes(3)�� MudOS v21c2�� (5 Sep 1994)

������:

�� write_bytes() - ����������д���ļ�.

���:

�� int write_bytes(string path, int start, string series);

�� ���� write_bytes(˵��ַ path, ���ʼ start, ˵��ַ series);

����:

�� write_bytes() ����������, ����ļ path ĵ start ֮'д��. ����ʱ�� 0, ��ɹ����ؽ 1.

����:

�� write_file(3), read_bytes(3)

������:

�� Spock @ FF 97.jul.25.�� (printed 3/16/95)

---

write_file(3)�� MudOS v21c2�� (5 Sep 1994)

������:

�� write_file() - appends a string to a file

���:

�� int write_file(string file, string str, int flag);

�� ���� write_file(̲��� file, ̲��� str, ���� flag);

���÷:

�� ������ str�� ̲ü̲Ļ file ��. ̲ı�� 1,
tʃ �� 0. ��� flag Ï 1, �� write_file() ̲Ķд (overwrite) file Ï
str, ̲Æ̲ str.

���o:

�� read_file(3) write_buffer(3) file_size(3)

������:

Spock @ FF
97.Jul.25.
(printed 3/16/95)

? ???? ???? ??????

??????????ęķ????ļ???

ص???hx

---

acos (3)�� MudOS v21c2�� (5 Sep 1994)

������:

�� acos() - ����h������������ (float) ķ������ (arccosine).

���:

�� float acos(float f);

�� ������ acos(������ f);

���-�:

�� ���ز��
ķ������, f ĵ�λ������
(radians).

���o:

�� asin(3), atan(3), cos(3), sin(3), tan(3)

������:

�� Spock @ FF 96.Oct.11.� (printed 3/16/95)

---

asin(3)���� MudOS v21c2�� (5 Sep 1994)

������:

���:

���÷:

����o:

������:

---

atan(3)���� MudOS v21c2�� (5 Sep 1994)

������:

�� atan() - ����h������������������ (tangent).

���:

�� float atan(float f);

�� ������ atan(������ f);

���÷�:

�� ���ز��ף f��
ķ�����, f ĵ̂λ�����
(radians).

���o:

�� acos(3) asin(3) cos(3) sin(3) tan(3)

������:

�� Spock @ FF 96.Oct.15.�� (printed 3/16/95)

ص���hx

---

ceil(3)���� MudOS v21c2�� (5 Sep 1994)

������:

���:

	�� float ceil(float f);

	�� ******* ceil(******* f);

���÷�:

	�� ****h****χ f�� ***ц**ъ**** f��
	**********, ** f**
	Cλ**********λ******.

***o*:

	** floor(3), to_int(3), to_float(3)

******:

	** Spock @ FF 96.Oct.24.** (printed 3/16/95)

---

cos(3)���� MudOS v21c2�� (5 Sep 1994)

������:

�� cos() - ����h������������������ (cosine)

���:

�� float cos(float f);

�� ������ cos(������ f);

���÷�:

�� ���ز���
f ������, f�� ����� (radians)
Ï��λ.

���o:

�� acos(3), asin(3), atan(3), sin(3), tan(3)

������:

�� Spock @ FF 97.Jan.27.�� (printed 3/16/95)

---

exp(3)���� MudOS v21c2�� (5 Sep 1994)

������:

�� exp() - ��h�������Ï¸��, ��Ƶ���� e�� Ï.

���:

```
�� float exp( float f );
```

```
�� ******* exp( ******* f );
```

���÷:

�� exp() ���� e^f ��.

���o:

�� log(3), pow(3), sqrt(3)

******:

�� Spock @ FF 97.Feb.12.�� (printed 3/16/95)

ص���hx

---

floatp(3)�� MudOS v21c2�� (5 Sep 1994)

������:

�� floatp() - ж̣ı̣p

���:

�� int floatp(mixed arg);

�� ���� floatp(��� arg);

���-�:

�� ��� arg ��ḥ̣̣̣̣̣̣̣̣̣ ̣ 1,
̣̣̣̣̣̣p̣̣̣ 0.

���o:

�� mapp(3), stringp(3), pointerp(3), objectp(3), intp(3), bufferp(3), functionp(3), nullp(3), undefinedp(3), errorp(3)

������:

�� Spock @ FF 97.Feb.12.

(printed 3/16/95)

---

floor(3)�� MudOS v21c2�� (5 Sep 1994)

������:

�� floor() - ��ц�� ��h��������,
��C�گ���ڈ:��������������, ��h��������������C��λ��ʒ.

���:

�� float floor(float f);

�� ������ floor(������ f);

���÷�:

�� ����h��� ������ f, ������ڈ�C�� f�� ������.

���o:

�� ceil(3), to_int(3), to_float(3)

������:

�� Spock @ FF 97.Feb.12.�� (printed 3/16/95)

---

log(3)���� MudOS v21c2�� (5 Sep 1994)

������:

���:

	�� float log(float f);

	�� ******* log(******* f);

���-�:

	�� ***ز*** f��
	****Ȼ****. f ****ï**.

******:

	�� exp(3), pow(3), sqrt(3)

******:

	�� Spock @ FF 97.Feb.18.�� (printed 3/16/95)

ص***hx

---

pow(3)���� MudOS v21c2�� (5 Sep 1994)

������:

���:

	�� float pow(float x, float y);

	�� ������ pow(������ x, ������ y);

���-�:

	�� pow() ���� x �� y ηΦ. �� x �� 0.0, y
	����ï����. �� x ï����, y ����ïh������.

���o:

	�� exp(3), log(3), sqrt(3)

������:

	�� Spock @ FF 97.Apr.24.�� (printed 3/16/95)

---

sin(3)���� MudOS v21c2�� (5 Sep 1994)

������:

�� sin() - ����h������������������ (sine).

���:

�� float sin(float f);

�� ������ sin(������ f);

���÷:

�� ���ز������������ f
ʃ̂λ����� (radians).

���o:

�� acos(3), asin(3), atan(3), cos(3), tan(3)

������:

�� Spock @ FF 97.Jul.23.�� (printed 3/16/95)

ص���hx

---

sqrt(3)���� MudOS v21c2�� (5 Sep 1994)

������:

���:

	float sqrt(float f);

	float sqrt(float f);

���÷:

	sqrt() ���ز��ڸ f ѕ����. f ѕ����ï��.

���o:

	exp(3), log(3), pow(3)

������:

	Spock @ FF 97.jul.23.�� (printed 3/16/95)

[ص]hx

---

tan(3)���h MudOS v21c2h (5 Sep 1994)

������:

h tan() - ����h������������������ (tangent)

���:

h float tan(float f);

h ������ tan(������ f);

���:

h ���ز��h f
������, f ʃ˙λ�����.
(radians).

���o:

h acos(3), asin(3), atan(3), cos(3), sin(3)

������:

h Spock @ FF 97.Jul.24.h (printed 3/16/95)

---

to_int(3)�� MudOS v21c2�� (5 Sep 1994)

������:

�� to_int - ��h������������ ��� (buffer) ⊓��Ïh�������.

���:

�� int to_int(float | buffer x);

�� ����� to_int(������ �� ������ x);

���-�:

�� ��� x�� ��h������������, �� to_int() ���� x�� C������ǰj������. ��� x�� ��h������������, �� to_int() ��� ������ǰ̃e������ (�� network-byte-order ������).

���o�:

�� to_float(3), read_buffer(3), sprintf(3), sscanf(3)

������:

�� Spock @ FF 97.Jul.25.�� (printed 3/16/95)

? ???? ???? ???? ????

??????????ęķ????ļ???

ص???hx

---

apply(3)�� MudOS v21c2�� (5 Sep 1994)

������:

�� apply() - ������ (function pointer).

����:

�� mixed apply(mixed f, mixed *args)

�� ��� apply(��� f, ��� *args)

���-�:

�� ��� f ��h������, ɔ���� f ������������ args ���� f IJ��� (argument).

�� ����:

���� apply(f, ({ 1, 2, 3 })); �� evaluate(f, 1, 2, 3);
�������.

������:

�� Spock @ FF���� 96.Oct.14.���� (printed 3/16/95)

bind(3)���� MudOS v21c2�� (5 Sep 1994)

������:

bind() - ���h������ (function pointer)�� ��Í���.

���:

function bind(function f, object ob)

bind(���f, ���ob)

���-�:

f Íob , ����Ǵ f ����.
f ĵ ��\ f ��h ��.
this_object
hЩ, bind() .

��:

```
void make_living(object ob) {
    function f;

    f = bind( (: enable_commands :), ob );

    evaluate(f);

}
```

ob
enable_commands() . !!
hЩєт õєќ, Ï bind()
h.

Ï��Ç��¨ bind() ������,
��h������������������ (master apply) valid_bind()
������������������ bind() ��������. ���� bind()
ƒ, �� valid_bind() ���� 1 ������, ���� 0
��� ¸.��.

�� Spock @ FF 96.Oct.16.�� (printed 3/16/95)

---

evaluate(3)�� MudOS v21c2�� (5 Sep 1994)

������:

�� evaluate() - ��h������ţ�� (function pointer)

���:

�� mixed evaluate(mixed f, ...)

�� ��� evaluate(��� f, ...)

���÷�:

�� ��� f�� ��h������,
ɔ��Ъ˩ţ�������ժ (������ ... ʔطʕ) ï f��
′ðIJ������ f. �� f�� ����ɜ���, evaluate(f, ...
)�� ��� ��ˌ̈ü�� f�� ʟ′��.

������:

�� Spock @ FF 97.Feb.12.�� (printed 3/16/95)

ص���hx

functionp(3)�� MudOS v21c2�� (5 Sep 1994)

������:

	�� functionp() - ж̣����ı����p̈ïḥ̈̈̈̈̈̈̈
	(function pointer). ����Ạ̊̈̈̈̈̈̈,
	̈̈̈̈̈ӝÏ̈̈̈̈̈̈̈̈̈̈.

���:

	�� int functionp(mixed arg);

	�� ���̈ functionp(̈̈̈ arg);

̈̈̈÷̈:

	̈̈ ̈̈̈ arg ̈̈ḥ̈̈̈̈̈̈̈̈̈ ط̣̈̈̈ (nonzero),
	̈̈̈̈̈̈̈̈ ̈ 0.
	̈̈ ̣̈̈̈̈̈̈̈̈ḧ̈̈̈̈̈̈̈̈(function)
	̈̈̈ ̣̈̈̈̈̈̈.
	̈̈ ̈̈̈̈̈̈ḧ̈̈̈̈̈:
	̈̈

		̈̈ f = (: obj, func :);

	̈̈
	̈̈̈ص̣̣̈̈̈̈̈̈̈̈̈̈̈̈̈̈̈,
	̈̈̈̈Щ̣̈̈̈̈̈̈̈̈̈̈̈̈̈̈̈̈̈̈̈̈̈̈̈̈̈ë
	"include/function.h"
	̈ļ̈̈̈.

	̈̈ function pointer typë̈ value
	̈̈ ----------
	̈̈ call_other̈̈̈̈ FP_CALL_OTHER
	̈̈ lfun̈̈̈̈̈̈ FP_LOCAL
	̈̈ efun̈̈̈̈̈̈̈̈̈̈FP_EFUN
	̈̈ simul_efun̈̈̈̈ FP_SIMUL
	̈̈ functional̈̈̈̈ FP_FUNCTIONAL

? ????? ????? h??

??????????ek?????l???

?ص???hx

---

map(3)���MudOS v21c2(5 Sep 1994)

������:

map() - ����h������ h��ÿ

���:

mapping map(mapping | mixed* | string x, string fun, object ob, mixed extra, ...);
mapping map(mapping | mixed* | string x, function f, mixed extra, ...);

ÿ map(ÿ * x, fun, ob, extra, ...);

ÿ map(ÿ * x, f, extra, ...);

���-�:

h h ÿ, map() map_mapping() ����.
h, map() map_array() .h, map()
e ÿh (character)
ժ ص ķ 滆 .

���o:

filter_array(3), sort_array(3), map_array(3), map_mapping(3)

������:

Spock @ FF 97.Feb.18. (printed 3/16/95)

nullp(3)�� MudOS v21c2�� (5 Sep 1994)

������:

�� nullp() - ж̣ḥp̣ị̈ (null).

���:

�� int nullp(mixed arg);

�� ���̣ nullp(̣ arg);

̣:

̣ arg ọ́иarg 1. arg
̣ị̈:

1. arg δ’ (initialize).

2. arg ḥψ̣ ̣.

3. arg ḥIJ̣,
ÖḥĞJ̣ƒIJ̣.

o:

mapp(3), stringp(3), pointerp(3), objectp(3), intp(3), bufferp(3), floatp(3), functionp(3), undefinedp(3), errorp(3)

������:

Spock @ FF 97.Apr.24.̣ (printed 3/16/95)

---

restore_variable(3) MudOS v21c2�� (5 Sep 1994)

������:

���:

	�� mixed restore_variable(string value);

	�� ��� restore_variable(?_��� value);

���÷�:

	�� ��h����_������ر���. ?_�����к?'��
	save_object() ?? restore_object() ??'?õ���.

���o?:

	?? restore_object(3)

������:

	?? Spock @ FF 97.Jun.2.?? (printed 3/16/95)

---

save_variable(3) MudOS v21c2�� (5 Sep 1994)

������:

���:

	�� string save_variable(mixed var);

	�� ����_ save_variable(��� var);

���÷�:

	�� ψ����_������h����_���.
	�������κ‘�� save/restore_object ����h��.

���o:

	�� save_object(3)

������:

	�� Spock @ FF 97.Jul.4.�� (printed 3/16/95)

---

sizeof(3)�� MudOS v21c2�� (5 Sep 1994)

����:

�� sizeof() - ����h������ (array)��ў�� (mapping)������ (buffer)������_��� (string) �� ����L.

���:

�� int sizeof(mixed var);

�� ���� sizeof(��� var);

���÷�:

�� ���� var �� ص���L, var ������Ѐ蘁 ���蘁ˍ������ǻ�. ��� var ������, ѝ�� 0.

���o�:

�� allocate(3), allocate_mapping(3), strlen(3)

������:

�� Spock @ FF 97. Jul.23.�� (printed 3/16/95)

typeof(3)�� MudOS v21c2�� (5 Sep 1994)

������:

�� typeof() - ����h��'�� (expression)
������������ (type).

���:

```
�� int typeof( mixed var );

�� ���� typeof( ��� var );
```

���-�:

�� ����h��'�ü�������������. <typeof.h> ж��屉
��Ï:

```
   ��
      ��  ������ INT�  2
      ��  ������ STRING  4
      ��  ���頹 ARRAY  8
      ��  ����  OBJECT  16
      ��  ÿ蜷 MAPPING  32
      ��  ������ FUNCTION 64
      ��  ������ FLOAT  128
      ��  ������ BUFFER  256
   ��
```

���o:

�� allocate(3), allocate_mapping(3), strlen(3)

������:

�� Spock @ FF 97.Jul.25. (printed 3/16/95)

undefinedp(3)�� MudOS v21c2�� (5 Sep 1994)

������:

�� undefinedp() - ж̣ı̣p̣δ.

���:

�� int undefinedp(mixed arg);

�� ���� undefinedp(��� arg);

���÷:

�� ��� arg�� ���ж���, �� 1.
�� arg ̣µ̣ị̈δ:

�� (1) ̣ï call_other
ص̣. (�� arg = call_other(obj, "???");)

�� (2) ̣ï
ị̈в̣ǰ̣.

���o:

�� mapp(3), stringp(3), pointerp(3), objectp(3), intp(3), bufferp(3), floatp(3), functionp(3), nullp(3), errorp(3)

������:

�� Spock @ FF 97.jul.25.
(printed 3/16/95)

? ???? ????
???????

??????????ek????ļ???

set_this_player.3
shout.3
snoop.3
this_interactive.3
this_player.3
userp.3
users.3
write.3

ص hx

---

add_action(3)�� MudOS v21c2�� (5 Sep 1994)

������:

��

add_action() - ��Ģ̄Ģ̄ (local function)
add_action() - ģ̄ģ̄ (local function)
add_action() - ģ̄ģ̄ (local function)
add_action() - ģ̄

輸ЧЦ h.

 <flag> 1 ,Ç ǰ (leading characters) ə <cmd> , (entire verb) query_verb() . 2 , ǰ, query_verb() <cmd> (characters following <cmd>).

o:

 query_verb(3), remove_action(3), init(4)

:

 Spock @ FF 96.Oct.11. (printed 3/16/95)

ص hx

---

command(3)�� MudOS v21c2�� (5 Sep 1994)

������:

�� command() - ����������, ��h�������.

���:

�� int command(string str, object ob);

�� ���� command(¸��� str, ��� ob);

���÷�:

0

����o�:

������:

Spock @ FF 97.Jan.27.

(printed 3/16/95)

commands(3)�� MudOS v21c2�� (5 Sep 1994)

������:

�� commands() -
����hЩЦ��� ���'ڑ���'õ�������ЩЈ

���:

�� mixed *commands(void);

�� ��� *commands(¤ ���);

���÷¤:

�� ����h������������������ ����� (array),
„�� this_object() ����'õĶ��� (action) .
��h������������������ (�� add_action()��
������) ; ���Ј������������ı¯ (flags) (��
add_action() ĵ̂������, Ц̨ïĬ 0��);��
Ни���„������.
DZ��„õĺ�� (function)
(Ѝ[function pointer]
¤ ¤ "<function>").

���o¤:

� add_action(3), enable_commands(3), disable_commands(3)

������:

� Spock @ FF 97.Jan.27. (printed 3/16/95)

---

disable_commands(3) MudOS v21c2�� (5 Sep 1994)

������:

���:

	�� int disable_commands(void);

	�� ���ǻ disable_commands(ǻ ��ǻ);

���÷:

	�ǻ ǻ�h������ŵ������g��ǻ��š(non-living).
	ǻ��v����ǻ��š, ǻǻǻ

disable_wizard(3) MudOS v21c2�� (5 Sep 1994)

������:

�� disable_wizard() - ą��h�������������ъ��Ę.

���:

�� void disable_wizard(void);

�� ���̡ disable_wizard(̈ �� ��);

���÷�:

�� ������������ enable_wizard() ������Ĺ���� .
��Ľǰ�������ъ��Ęą����.

���o:

�� enable_wizard(3), wizardp(3)
��

������:

�� Spock @ FF 97.Feb.04.�� (printed 3/16/95)

---

ed(3)���� MudOS v21c2�� (5 Sep 1994)

������:

	�� ed() - �� h����ļ.

���:

	�� void ed(string file, string exit_fn, int restricted);
	�� void ed(string file, string write_fn, string exit_fn, int restricted);

	�� �� ���˛ed(
	˛���˛ file, ˛���˛ exit_fn, ˛���˛
	restricted);
	�� �� ���˛
	ed(˛���˛ file, ˛���˛ write_fn, ˛���˛ exit_fn, ˛���˛
	restricted);

���÷�:

	�� ˛˛˛

���

int write_fn(string fname, int flag)

�� �� �� ����ħ, MUD ������ (driver) ����
exit_fn ����. ������õÍ������ mudlib
���. ������Ç������µк'':
��

���

void exit_fn()
�� ������?Ъ restricted ¬
�����ʊ ����Ĺ���. ����: ����h���ļ,
Ғ������ļ. ЩЦ������ restricted ¬
Ï���根ħ, ��� '��.

���o:

regexp(3), valid_read(4), valid_write(4), get_save_file_name(4)

������:

Spock @ FF 97.Feb.7.

(printed 3/16/95)

---

enable_commands(3) MudOS v21c2�� (5 Sep 1994)

������:

�� enable_commands() - ��������'������ (player) ������.

���:

�� void enable_commands(void);

�� �� ���¸ enable_commands(�� ��);

���÷�:

�� enable_commands() �� this_object()�� ���ï���� (living) ������, ����������'�� add_action() ������������ (Ç'�� command() ����) . ���� enable_commands() ʰ, MUD�� �������� (driver) ��Ψ����گ��� catch_tell() ����. ����ц��،'���, ÿh������Ш������h������ʰ (���� say()) , ��������ɔ���� catch_tell() h��.
��

����¸������:

�� ������ create() ������ʲط����� enable_commands() , ����������¸�����.

���o�:

�� this_object(3), living(3), add_action(3), command(3), catch_tell(4), say(3), create(4)

������:

�� Spock @ FF 97.Feb.12.�� (printed 3/16/95)

exec(3)���� MudOS v21c2�� (5 Sep 1994)

������:

�� exec() -�� ��h����į������ (interactive) ��� (player) ����л������ (connection) ��������h�������.

���:

```
�� int exec( object to, object from );

�� ���� exec( ��� to, ��� from );
```

���÷�:

```
��
������������������ĵ������ŵį�������
(link) Ǩ�Ƶ�����h�������. Xį����ٮ, ��ı����ę
exec( to, from ) ȩ, interactive( to ) ��� 1, ��
interactive( from ) ��� 0. ������ from ����h�ת�p���Ę
to ������.

��
ĕĕ̵ĕĕĕĕĕĕĕĕĕĕĕĕĘ̆ĕĕĕdzĕĕĕ,
������'ê̂û̂Σ̂ MUD̂ ̂ ̂єтĕĕõ.
ĕĕĵĕĕĕĕĕ'ê̂ô̂'ĕĕĕĕķĕĕĕĕĕĕĕĕĕдhĕĕ¤ĕĕĕ
(simulated emulated function, simul_efun), ĕĕ'ĕĕ
valid_override(4)ĕĕ ĕĕĕĕĕy'ê̂hġĕĕĕ  ĕĕĕĕĕĕ'ĕĕ.
(ê̂
efun::exec()
)

ĕĕ exec() ĕĕĕĕĕگıĕĕĕת̂ĕĕ  ê̂ 1, ת̂ĕĕĵĕĕĕ  ê̂ 0.
```

ĕĕĕo̊:

ĕĕ interactive(3), valid_override(4)

find_player(3)�� MudOS v21c2�� (5 Sep 1994)

������:

�� find_player() - ������Ψ��h�����

���:

�� object find_player(string str);

�� ��� find_player(̲��� str);

���÷:

�� ̲ find_living()̲ ̲̲̲̲, ̲̲̲̲Ψ̲e̲ı̲̲̲̲̲ (interactive) ̲̲̲̲̲̲̲̲ï̲ı̲̲̲̲̲̲̲̲.

���o:

�� lower_case(3), find_living(3), livings(3), users(3), set_living_name(3)

������:

�� Spock @ FF 97.Feb.12.�� (printed 3/16/95)

ص���hx

get_char(3)�� MudOS v21c2�� (5 Sep 1994)

������:

�� get_char - ��ˇ������̣�����h��̣���Í���.

���:

�� varargs void get_char(string | function fun, int flag, ...);

�� ������ ̣ ̣ ̣

in_edit(3)�� MudOS v21c2�� (5 Sep 1994)

������:

�� in_edit() -
  ж h p .

���:

```
�� string in_edit( object default : F_THIS_OBJECT );
```

```
�� in_edit( ��  : F_THIS_OBJECT );
```

���-�:

```
�� ���h                                  ,
  и  ,  .  ,  0.
```

���o:

�� ed(3), in_input(3)

������:

�� Spock @ FF 97.Feb.14.�� (printed 3/16/95)

in_input(3)�� MudOS v21c2�� (5 Sep 1994)

������:

�� in_input() - ж h p input_to ġ‘

���:

```
�� int in_input( object default : F_THIS_OBJECT );

�� ���  in_input( ��  Ц  : F_THIS_OBJECT );
```

���-�:

�� input_to get_char ġ‘, , 1.

���o�:

�� get_char(3), input_to(3)

������:

�� Spock @ FF 97.Feb.14.�� (printed 3/16/95)

---

input_to(3)�� MudOS v21c2�� (5 Sep 1994)

������:

�� input_to() -
����h������������������h��ۻ���Í���

���:

�� varargs void input_to(string | function fun, int flag, ...);

�� ������ ۻ ۻ ۻ
input_to(ۻۻ ۻ ۻ fun, ۻۻۻ
flag, ...);

���÷�:

��
��'������h������������Ï����ۻ fun
IJۻ. ۻۻ ۻ ۻ (driver)
ۻۻ (parse).

�� ۻ۸ۻ,
input_toۻ ãۻۻ۸ۻ۸ъۻ, ۻ ۻۻ input_to
ۻۻۻۻۻۻۻۻۻۻۻۻۻۻۻۻ.
ۻۻۻۻۻۻۻۻۻۻ input_toۻ۸ ۻۻۻкγۻۻۻۻ.
ۻۻ'ۻۻۻۻۻδۻۻۻۻۻ, ۻۻۻۻۻۻۻۻۻ
fun.

ۻۻ ۻۻۻhۻۻۻۻۻۻۻ input_to()
ۻۻۻۻhۻۻۻۻۻ, ۻeۻhۻ۸εۻۻۻۻЧ.

ۻۻ ۻۻۻۻۻ flag Ïۻۻۻۻ,
ۻۻۻۻۻۻۻۻۻۻۻۻ ۻۻۻۻÖ (echo) ۻ۸ۻۻۻۻۻ,
Χۻۻۻۻۻۻۻ۸ۻۻۻ (snoop)ۻۻ ۻĻۻۻۻۻ
(ۻۻۻۻۻۻۻۻۻ۸ħۻۻۻۻۻۻۻ).

ۻۻ 'ۻۻۻۻۻۻۻۻhۻۻۻۻۻ ۻۻۻ۸ۻۻ۸úۻۻۻۻۻ

fun`` ﬁ`ĵ`h`````` (````ï`_```).`` `` flag
˜``IJ````p`` fun``
`ĵ J`````````````.....`ŋŋIJ```.

````o`:

`` call_other(3), call_out(3), get_char(3)

``````:

`` Spock @ FF 97.Feb.14.`` (printed 3/16/95)

---

`ص```hx

---

# interactive(3)�� MudOS v21c2�� (5 Sep 1994)

## ����:

�� interactive() - ж h p ї ц g (interactive)

## ���:

```
int interactive( object ob );
```

```
int interactive( ob );
```

## ���-:

ob h ц , и ط . Ч (link dead), 0.

## ...:

query_ip_number(3), query_ip_name(3), enable_commands(3)

## ......:

Spock @ FF 97.Feb.14. (printed 3/16/95)

# message(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� message() - ����Ш������ŵ (living) ���.

## ���:

�� void message( mixed class, string message, mixed target, mixed exclude );

�� ¦ ¦¦¦¦
message( ¦¦¦ class, ¦¦¦¦ message, ¦¦¦ target, ¦¦¦ exclude );

## ���÷¦:

�� message() ����L¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦
(¦¦¦¦ exclude ¦¦¦¦¦¦¦¦¦) ¦¦ receive_message( ¦¦¦ class, ¦¦¦¦ message ) ¦¦¦¦.
¦¦¦¦¦¦¦¦¦¦¦Ñ¦¦ ¦¦¦¦h¦¦¦¦Ш.
¦¦
¦¦ class ¦Ǵ¦¦¦Ш¦¦¦¦¦¦¦,
¦¦¦ڳ¦¦¦¦¦Ш¦Ľ¦¦¦¦¦¦ڑ¦. ¦¦¦¦ combat, shout, emergency ¦ŋ¦.

¦¦ message ¦¦¦¦¦ד¦¦¦¦¦Ш¦¦¦¦.

¦¦ target ¦¦¦¦¦¦¦¦¦Ш¦¦¦¦¦¦¦¦¦¦,
¦¦¦¦¦¦h¦¦¦¦¦¦¦¦¦¦¦¦¦¦ (pointer).
¦¦¦Ľ¦¦¦¦h¦¦¦å¦¦ŵ (non-living) ¦¦¦¦¦,
¦J̌¦¦¦¦¦¦ڳ¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦ɻ¦¦¦Ш.

¦¦ exclude ¦Dz¦Ö¦ɻ¦¦¦¦¦Ш¦¦¦¦¦¦¦¦¦¦,
¦¦¦¦¦¦h¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦ (array).

## ¦¦¦o¦:

•• say(3), write(3), shout(3), tell_object(3), tell_room(3)


••••••:

•• Spock @ FF 97.Apr.15.•• (printed 3/16/95)

---


ص•••hx


---

# notify_fail(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

	�� notify_fail() - ����h�δ������Ш.

## ���:

	�� int notify_fail( string | function str );

	�� ���� notify_fail( ۪��� ۪۪ ۪۪۪۪۪ str );

## ���÷�:

	۪۪ ۪۪ str۪۪
	۪۪۪єт۪۪۪۪۪۪۪۪۪Ч۪۪۪ص۪Ј۪۪۪۪۪Ш :۪۪۪ô ?
	۪۪۪۪ add_action() ۪趨۪Ķ۪۪۪۪۪۪۪ 0۪۪ ۪,
	۪ɔ۪۪۪'۪„۪ str ۪۪۪۪Ш. '۪۪ notify_fail()۪۪
	۪۪'۪۪۪۪۪۪Ш۪۪h۪۪۪ŁМõķ۪۪۪,
	۪۪Ї ¨ ۪h۪۪۪۪۪۪۪۪۪

********:

** notify_fail() **2*****  0.

**********:

** Spock @ FF 97.Apr.24.** (printed 3/16/95)

---

**ص***hx

---

# printf(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� printf, sprintf - ת������̣����‘��������.

## ���:

```
�� void printf( string format, ... );
�� string sprintf( string format, ... );
```

```
�� ̄  ���̣ printf( ̄��  format, ... );
�  ̄��  sprintf( ̄��  format, ... );
```

## ���÷:

```
   ̄ ̄ ̄Ï LPC (s)printf() ̄к‘÷, Lynscar (Sean A Reith)
   ̄ ̄ ̄hЩ̄ ̄ ̄÷.

   Ьǰ̄Ì汾  ̄ ̄µ  к‘ (modifier):
```

```
      " " ̄ ̄ ̄ ̄ ̄ ̄ǰ ̄ ̄ ̄h ̄ ̄n.

      "+" ̄ ̄ ̄ ̄ ̄ ̄ǰ ̄ ̄ ̄h +  .

      "-" ̄ ̄ ̄ ̄λ (field)  ̄ ̄ ̄ ̄.

            ̄ ̄ ̄ ̄    :  ̄ ̄ ̄ ̄
            (s)printf()
            ̄ ̄ ̄ ̄Ч ̄ ̄ ̄ ̄ ̄ ̄
```

---



---

# query_host_name(3) MudOS v21c2�� (5 Sep 1994)

## ������:

�� query_host_name() - ����'�õ������������ (host name)

## ���:

�� string query_host_name( void );

�� ̲���� query_host_name( ̲ ̲ ̲

# query_idle(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� query_idle()�� - ��ğh������������ (interactive) ����ĞŌ����� (idle) ����.

## ���:

```
�� int query_idle( object ob );

�� ���� query_idle( ��� ob );
```

## ���÷�:

```
�� ��ğh������������ (interactive)
   ��������ĞŌ����� (idle) ����.
```

## ���o�:


�� in_edit(3) in_input(3)


## ������:


�� Spock @ FF 97.May.1.�� (printed 3/16/95)

# query_ip_name(3) MudOS v21c2�� (5 Sep 1994)

---

## ������:



## ���:

	�� string query_ip_name( object ob );

	�� ��ц��ц query_ip_name( ��ц ob );

## ���ц:

	�� ������ц ob ц IP ц. h������ц addr_server ц.ц.ц.

# query_ip_number(3) MudOS v21c2�� (5 Sep 1994)

---



---



---

# query_snoop(3)�� MudOS v21c2 (5 Sep 1994)

---

## ������:

�� query_snoop() - ����=���ټ������Ļ��
(interactive) ���.

## ���:

```
�� object query_snoop( object ob );

�� ��� query_snoop( ��� ob );
```

## ���÷:

```
�� ������������ ob
������h������������������,
и��  ��������. ���û�м�����, ��  0.
```

## ���о:


�� snoop(3), query_snooping(3)


## ������:


�� Spock @ FF 97.May.1.�� (printed 3/16/95)


---



---

# query_snooping(3) MudOS v21c2�� (5 Sep 1994)

---

## ������:

�� query_snooping() - ��������ي��ه (snoop) ������.

## ���:

�� object query_snooping( object ob );

�� ��ه query_snooping( ��ه ob );

## ���ـ:

�� ������ه (interactive)�ه ��ه ob
���ي�ه؟h
ر ّ. ه ob ûم
0.

## ���o:


�ه snoop(3), query_snoop(3)


## ������:


Spock @ FF 97.May.1 (printed 3/16/95)


---

# receive(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� receive() - ��Ľǰ�������’h����Ш.

## ���:

�� int receive( string message );

�� ���� receive( ˍ���  message );

## ���-�:

# remove_action(3) MudOS v21c2�� (5 Sep 1994)

## ������:

�� remove_action - ��X_����� (local function)
�н��h�����  ?? (command verb)

## ???:

?? int remove_action( string fun, string cmd );

?? ???? remove_action( ?_???? fun, ?_???? cmd );

## ???-?:

?? remove_action(3)
??h?????????_?????????н??h?????  ??.
?? ????????, remove_action() ????????? add_action(3)
?? add_verb(3) ?Í????. ??????????Çh????????ñ,
?J?????? remove_action() ???.

## ???????:

?? remove_action() ?k???:

?? 1 ????????ı?.

?? 0 ????ƒ??.

## ????o?:

?? add_action(3), query_verb(3), init(4)

## ???????:

?? Spock @ FF 97.May.27.?? (printed 3/16/95)

# resolve(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� resolve() - п��h������������·��_(internat address)
Ї�������������(domain name), ����ℭ..

## ���:

�� int resolve( string address, string callback_func );

�� ����resolve( ?_���address, ?_���callback_func );

## ���÷?:

�� resolve() ��140.128.136.11" ����п" ��'�� address��
��Ї "bbs.csmc.edu.tw" ������'������������.
���ў "bbs.csmc.edu.tw" ������'�address��
140.128.136.11" ����п"�� ��'?ĵ?_ п���ı���,
������Lў������ callback_func. callback_func ?к?'Ї:

�� void callback(string address, string resolved, int key);

?? ?  ???_callback(
?_???address, ?_???resolved, ????
key );

?? key ?? resolve() ?ķ???_address
??п????ĵ?_??????. ???п??ų??, ???? 0.

## ???o?:

?? query_host_name(3), socket_address(3), query_ip_name(3),
query_ip_number(3)

## ������:

?? Spock @ FF 97.Jun.2.?? (printed 3/16/95)

# say(3)���� MudOS v21c2�� (5 Sep 1994)

## ������:

�� say() -
����Ш��  ��������h��������e�'����.

## ���:

�� varargs void say( string str, object obj );

�� ������ ��  ���_say(
�_��  str, ��  obj );

## ���÷�:

��
����Ш��  ��λ��Ј������Ш��h
��Ł�����Ј̊����Ш.
������Ш��_���  this_player(), ���ă this_player() ==
0 ��  this_object().

�� ��Ј.
����_���~Ј������  obj, ����Ш���л  ��
obj.  ���  obj  ����h,
����h  (an array of objects) ,
����Ш�Т��л��Ј������  ����.

## ���o:


�� message(3), write(3), shout(3), tell_object(3), tell_room(3)


## ������:


�� Spock @ FF 97.Jul.4.�� (printed 3/16/95)

# set_this_player(3) MudOS v21c2�� (5 Sep 1994)

---

## ������:

	�� void set_this_player( object who );

	�� ��  ���_set_this_player( ��� who );

## ���:

	�� set_this_player() '��� who�� ����µ this_player().
	�� ����  ���� (efun)¸ ���Ĵ���
	__NO_ADD_ACTION__ ��  mudlib��
	���¸'������ĥ'���  __NO_ADD_ACTION__
	��������²���Ч.

	�� who ��������, ��  this_player() ���趨Ï��.

## ���o:

	��  this_player(3)

## ������:

	��  Spock @ FF 97.Jul.21.��  (printed 3/16/95)

---



---

# shout(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� shout() - ������л��ŵ (living) ����т���Ш.

## ���:

```
�� void shout( string str );

��  ��  ��_shout( ��_��  str );
```

## ���-�:

```
�� ���  this_player() ����,
      ������л��ŵ�����т��_���  str.
```

## ���o:


�� message(3), write(3), tell_object(3), tell_room(3), say(3)


## ������:


�� Spock @ FF 97.Jul.23.�� (printed 3/16/95)

# snoop(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� snoop() - ����h���.ţ�����'���� (interactive user).

## ���:

�� varargs object snoop( object snooper, object snoopee );

�� ������ ��ŏ snoop( ��ŏ snooper, ��ŏ snoopee );

## ���÷�:

�� ������������ĥ, ŏ snooper ŏ'���ŏ snoopee. ������ŏ ĵ������, ŏ'snooper ���eĿ���ˮ. ć����'ŏ simul_efun ŏ ŏï���ŏ snoop() ŏi ő ŏfŋ. ������������ĥ, snoop() ŏ.ı���ŏ snoopee. ŏh������ĥ, ŏ ŏ snooper. ����ŏ 0ŏ ŏ'ŋŏ.

## ���o:

�ŏ query_snoop(3), query_snooping(3)

## ������:

ŏ Spock @ FF 97.Jul.23.ŏ (printed 3/16/95)

---

# this_interactive(3) MudOS v21c2�� (5 Sep 1994)

## ������:

�� this_interactive - ���ص��ô'����������.

## ���:

�� object this_interactive();

�� ��� this_interactive();

## ���÷�:

��
���ص��ô'����������.
������ this_interactive() ��� this_player(), ������ this_player() 	 	 	

# this_player(3)�� MudOS v21c2�� (5 Sep 1994)

---

## ���:

�� object this_player( int flag );

�� ��� this_player( ���� flag );

## ���-�:

�� ���ص��ô'������������ (player) ���. ��ש��,
��'ĠĠ������������گ�'��
this_player(), ��ֵ��Ẋ������ this_object() ��ʳ. ������
this_player(1) ����,
ص��ô'����Ŀ���
(interactive)�� ���. this_player(1)��
ķ������ijЩĮ������������ this_player() ��ʳ. ����:
єҨ����Ų (admin)�� '�� command()��
ǿ��h������������hЩĮ����ĥ.

## ���o�:

�� this_object(3)

## ������:

�� Spock @ FF 97.Jul.25.�� (printed 3/16/95)

---



---

# userp(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� userp() - ж.p(ї)ц(interactive).

## ���:

```
int userp( object );
```

```
userp( ��( ) );
```

## ���-:

arg ( ) ,  1.

## ...o:

interactive(3), users(3), living(3)

## ������:

Spock @ FF 97.jul.25. (printed 3/16/95)

# users(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� users() - ����h������������ц��  (interactive) ��ц�����.

## ���:

```
�� object *users( void );

�� ��  *users( ?   ??  );
```

## ���-?:

```
?? ????h??????????,
??????????????eL.ц?????.
```

## ???o?:

?? livings(3), children(3), objects(3)

## ������:

?? Spock @ FF 97.jul.25.?? (printed 3/16/95)

---

# write(3)�� MusOS v21c2�� (5 Sep 1994)

## ������:

�� write() - ��Lў������т���Ш.

## ���:

�� void write( mixed str );

�� ��  ���_write( ��� str );

## ���-�:

�� ����Ш str�� ��Lў�����. str
������h������,
������_п����_����т�.

## ���o:

�� message(3), tell_object(3), tell_room(3), shout(3), say(3)

## ������:

�� Spock @ FF 97.jul.25.� (printed 3/16/95)

# ? ???? ???? ?گ?

## ??????????ęķ????į???

# cache_stats(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� cache_stats() - ������������ (driver) ��  mud ������ (mudlib) ѵ仸ҡ������.

## ���:

�� void cache_stats( void );

�� ѵ  ���ˌ cache_stats( void );

## ���÷�:

��
������ۍ۔ۓ۔ۓ۔۔۔۔۔۔۔۔۔۔ۓۓۓ

# debug_info(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� debug_info() - ��'���� (debug) ��Ш

## ���:

```
�� mixed debug_info( int operation, ... );
�� mixed debug_info( 0, object ob );
�� mixed debug_info( 1, object ob );

�� ��� debug_info( ���� operation, ... );
�� ��� debug_info( 0, ��� ob );
�� ��� debug_info( 1, ��� ob );
```

## ���-�:

```
�� debug_info()�� ��h������ө;ĵ�fŋ, ����Ï MudOS
����������.
�������Ç�ij�����ШЈ��ө�h.
����IJ�������ө�h������Ķ���
(operation)�� ����.

�� ���ĴĶ��� (0 ~ 1)
��Ç�Ĵ.
MudOS ����ű�IJ#��X ( various fields of the MudOS object
structure).

�� ���µ LPC
������h���  ������:

�� ����h:

    /* di0.c */
    create() {
        debug_info(0, this_object());
```

```
        ���� }
�� ���h:
        O_HEART_BEAT��������������: FALSE
        O_IS_WIZARD������������������: FALSE
        O_ENABLE_COMMANDS : FALSE
        O_CLONE�������������������������: FALSE
        O_DESTRUCTED����������: FALSE
        O_SWAPPED��������������������: FALSE
        O_ONCE_INTERACTIVE: FALSE
        O_RESET_STATE���������: FALSE
        O_WILL_CLEAN_UP����: FALSE
        O_WILL_RESET: TRUE
        total light : 0
        next_reset��: 720300560
        time_of_ref : 720299416
        ref������������������: 2
        swap_num�������: -1
        name������������������: 'u/c/cynosure/di0'
        next_all�������: OBJ(bin/dev/_update)
        This object is the head of the object list.

�� ������:

    ��/* di1.c */
        ����create() {
            ������debug_info(1, this_object());
        ����}
�� ������:
        program ref's 1
        Name u/c/cynosure/di1.c
        program size 10
        num func's 1 (16)
        num strings 0
        num vars 0 (0)
        num inherits 0 (0)
        total size 104
��
```

## ����o�:


•• dump_file_descriptors(3) dump_socket_status(3)


## ������:

•• Spock @ FF 97.Jan.27.•• (printed 3/16/95)

---


ص���hx


---

# debugmalloc(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� debugmalloc() - ��������õ��ڴ�(malloc'd memory)
��Ш��з��h����¦���.

## ���:

�� void debugmalloc( string filename, int mask );

�� ¬ ��¬ debugmalloc( ¬���
filename, ¬��¬ mask );

## ���÷:

�� ¬¬¬¬ ¬¬¬¬(efun)
¬¬¬¬¬ڴ¬¬¬¬¬¬¬¬¬¬¬¬(driver)ʃ, ¬¬ options.h¬¬
ʃh¬¬¬¬ DEBUGMALLOC ¬¬
DEBUGMALLOC_EXTENSIONS ¬¬¬Ч. debugmalloc()
¬¬¬¬з DMALLOC()¬¬
¬¬¬õ¬¬ڴ¬¬¬¬¬¬¬¬Ш¬¬¬¬ص macros. (¬¬¬
macro ¬¬ mask bitwise and'd (&) with the tag, ¬¬ mask & tag¬¬
¬¬¬¬¬¬)
¬¬þ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬' ¬¬¬e¬ md.c ¬¬
config.h

## ¬¬¬o¬:

¬¬ set_malloc_mask(3)

## ������:

¬¬ Spock @ FF 97.Jul.23.¬¬ (printed 3/16/95)

# dump_file_descriptors(3) MudOS v21c2�� (5 Sep 1994)

## ������ц

�� dump_file_descriptors() - ��ʒ MudOS ������ļ������

## ������

�� void dump_file_descriptors( void );

�� ��  ���̮ dump_file_descriptors( ��  ��� );

## ���÷���

�� ��ş������������Ї̮ ��ˆ ��ˆ MudOS ������ij���J˜���hЦЩδ��ş��Ч���dz��ڳ��UN*X ��цєт�����єIJ������ˆ̃̃̃

## ����o���



## ⁄⁄⁄⁄⁄


   ⁄⁄ Kenny@Broken.History⁄⁄ 97.Jul.26⁄⁄ (printed 3/16/95)
   ⁄⁄


---



---

# dump_prog(3)�� MudOS v21c2���� (5 Sep 1994)

## ������ц

	�� dump_prog() - ��з/�������h�� LPC ���
	��

## ��  ⌐  ��

	�� void dump_prog( object ob, int flags default: 0, string file );

	��  ⌐  ���dump_prog(
	��⌐ ob, ⌐⌐⌐⌐ flags Ц��: 0, ⌐⌐⌐⌐
	file );
	⌐⌐

## ⌐⌐ ⌐÷⌐⌐⌐

	⌐⌐ dump_prog() ⌐⌐з⌐⌐⌐گ⌐⌐⌐ obj
	⌐⌐⌐⌐Ш⌐⌐ij⌐⌐⌐ļ⌐ `file'⌐⌐⌐⌐û⌐⌐⌐⌐ `file'
	⌐Ļ⌐⌐⌐⌐⌐⌐Ј⌐⌐⌐
	"/PROG_DUMP"⌐⌐⌐⌐⌐⌐⌐⌐⌐⌐⌐⌐⌐⌐⌐⌐⌐⌐⌐⌐⌐⌐⌐⌐û⌐⌐⌐◂

	⌐⌐ flags ⌐⌐⌐⌐⌐⌐⌐⌐⌐⌐⌐⌐⌐⌐⌐⌐⌐⌐⌐

# dump_socket_status(3) MudOS v21c2�� (5 Sep 1994)

## ������:

	�� dump_socket_status() - ̈r̈ÿḧ̈ LPC ̈̈ socket ̆″

## ���:

	̈̈ void dump_socket_status( void );

	̈̈ ̈  ̈̈̈ dump_socket_socket( ̈  ̈̈̈ );

## ���÷̈:

	̈̈ dump_socket_status()̈̈ ̈̈ḧ̈̈̈ĕĹ̈̈̈, ̈r̈ MudOS̈̈ ̈̈̈̈̈̈̈̈̈̈̆̈̈̈ LPC socket ̈̈Ŀĭ″. ̈’̈̈̈̈̈̈̈̈̈ LPC sockeẗ̈ Ö̈ó̈̈̈ (application)̈̈ ̈ij̈̈̈. ̈̈’̈̈ÿḧӥ̈̈̈̈ḧ̈ LPC socket.̈̈ ̈̈ḧ̈̈̈ LPC socket ̈ĵ̈ḧ̈̈̈̈̈̈ (descriptor), ̈Ĵ̈̈̈̈ģĴ̈̈̈̈̈̈̈, ̈Ղ̈̈̈̈̈̈. sockeẗ̈ ̈̈̈̈̈̈̈̈ӟ̈̈̈̈̈̈̈̈̈̈̈̈̈̈ʰ̈Ԍ̈̈̈.

	̈̈ ̈̈ḧ̈ Fd ̈̈̈̈ LPC socket ̈̈ص̈̈̈ɥɛҭ̈̈̈ļ̈̈̈̈̈̈. State ̈̈Ŀĭ̈̈ LPC socket ̈̈̈̈̈̈̈̈̈. Modë̈ ̈̈ socket ģ‘ (mode) , ̈̈ socket_create()̈̈ ̈IJ̈̈̈̆h. Local ̈̈ Remote address̈̈ ̈̈ Internet λ̈̈̈̈̈̈ў̈ (port number), ̈̈ Internet ̈ķ̈̈̈’̈̈ (dot notation, ̈̈ aaa.bbb.ccc.ddd ̈к̈‘) ̈̈̈’. * ̈̈̈’ḧ̈λ̈̈̈̈Ñô̈̈̈̈ï̈̈. ̈̈ʉ̈̈, CLOSED̈̈ ̆″̈̈ LPC socket Ï̈̈δ′̈ṏ socket, ̈̈̈̈Ը̈ sockeẗ̈ ̈̈’̈̈̈̈̈̈ ̈̈̈̈̈̈.

	̈̈ ̈̈̈µ̈̈̈̈̈̈̈̈̈̈̈̈ Portals mud̈̈ ̈̈̈̈, ̈̈ʰΨḧ̈ socket Ö̈ó̈̈̈ (application)̈̈ Ï MWHOD. ̈̈̈̈̈̈̈’Ŀĭ̈̈̈̈’̈̈̈̈̈̈̈ socket.̈̈ ḧ̈̈̈ STREAM ģ‘̈̈ socket ̈̈̈̈̈̈̈̈̈̈̈̈̈̈̈Ç̈̈.
	̈̈ḧ̈̈̈ DATAGRAM ģ‘̈̈ socket

### ĵ¸

| Fd | State | Mode | Local Address | Remote Address |
|---|---|---|---|---|
| 13 | LISTEN | STREAM | *.6889 | *.* |
| 14 | BOUND | DATAGRAM | *.6888 | *.* |
| -1 | CLOSED | MUD | *.* | *.* |
| -1 | CLOSED | MUD | *.* | *.* |

### o:

- debug_info(3), dump_file_descriptors(3)

### :

- Spock @ FF 97.Aug.22. (printed 3/16/95)

---

hx

---

# dumpallobj(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� dumpallobj() - ������������ë�����hЩт������.

## ���:

�� void dumpallobj( string | void );

�� ¬ ���¬ dumpallobj( ¬���¬ ¬¬ ¬ ¬ ¬¬¬ );

## ���÷¬:

��
������������з¬¬¬¬¬¬¬Ґ¬¬¬¬¬ë¬¬¬¬б¬.
¬¬¬û¬¬¬¬¬¬¬¬, ¬¬Щ¬¬Щ¬¬¬¬з¬¬
/OBJ_DUMP¬¬ ¬¬¬¬¬¬¬. ¬¬¬¬¬¬¬¬¬¬¬,
¬¬¬¬¬¬¬¬¬¬¬¬Ç¬¬¬¬¬¬¬¬¬¬¬¬.

## ¬¬¬o¬:

¬¬ mud_status(3), debug_info(3)
¬¬

## ¬¬¬¬¬¬:

¬¬ Spock @ FF 97.Feb.04.¬¬ (printed 3/16/95)

# get_config(3)�� MudOS v21c2�� (5 Sep 1994)

---

## ������:

�� get_config() - ��ğhЩ������������ (driver) ��єт趨 (config settings)

## ���:

```
�� string | int get_config( int );

�� ·���· ·· ····· get_config( ····· );
```

## ���÷:

```
·· ····· ····· (efun)
·····ğhЩ·····єт趨. ··o·
runtime_config.h ···_·Lǰ····Щ·····′·õ·w·····L.
```

## ···o·:

```
·· /include/runtime_config.h
```

## ������:

```
·· Spock @ FF 97.Feb.14.·· (printed 3/16/95)
```

---

# malloc_status(3) MudOS v21c2�� (5 Sep 1994)

## ������:

�� malloc_status() - ����hЩЦ�ڲ'������ε�Т������

## ����:

```
�� void malloc_status( void );

��  ��  ��¸ malloc_status( ¦   ��¦ );
```

## ���÷�:

```
��
������������ڲ'������ε�Т�������'ЎЈ
������Ґ֔ę���  vanilla 3.1.2��  ��
malloc��
����. ��ע��,
malloc_status() ����L'��, ������������ħ,
options.h
ўж������ڲ'''''''''''''''''''''.
```

## ���o:

```
�� mud_status(3), dumpallobj(3), memory_info(3)
```

## ������:

```
�� Spock @ FF 97.Feb.18.��  (printed 3/16/95)
```

# memory_info(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� memory_info - ḍ��h������������ mud ���ڳ'����.

## ���:

�� varargs int memory_info( object ob );

�� ������ ���ڳ memory_info( ��ڳ ob );

## ���÷:

�ڳ ����ڳ ob,ڳ ڳ memory_info()ڳ ڳ ob ڳ'õ��ڳ. ڳû۪ڳ, ڳ ڳ mud ڳ ڳ'ڳ, ڳ mud ڳ ڳ'ڳ ڳ ڳ

# moncontrol(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� moncontrol() - ������ر[д (profiling).

## ���:

�� void moncontrol( int on );

�� ��  ���¸ moncontrol( ���� on );

## ���÷:

�� ������ on Ї 1, moncontrol()�� ЈJ����д. ��� on ��Ї 0�� , moncontrol()�� ��ر[д. ������������ (driver) ћ������ћ���ј[д, ���¸ ��÷������ёдijЩ�����e�����. ������������������ (compile) ћ, û[п����д����, moncontrol() ��û��Ч��.

## ����o:

�� opcprof(3), function_profile(3)

## ������:

�� Spock @ FF 97.Apr.15.� (printed 3/16/95)

# mud_status(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� mud_status() -�� ����hIIЦ������ (driver) ��
mud ������ (mudlib)�� ��т������.

## ���:

�� void mud_status( int extra );

�� ��  ���¸ mud_status( ���� extra );

## ���÷�:

�� ��������hIIЦ������ mud��
������т������'ЎЈ
������ extra�� Ї����,
�����'����к������. ������ďЏ����
vanilla 3.1.2 汾e status ��status tables ��.

## ���o:

�� debug_info(3), dumpallobj(3), memory_info(3), uptime(3)

## ������:

�� Spock @ FF 97.Apr.15.�� (printed 3/16/95)

---

# opcprof(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� opcprof() - ����hЦ¶ ���� (efuns) ������õ·Ƶ��.

## ���:

�� void opcprof( string | void );

�� ¶ ���¸ opcprof( ¶¸��¸ ¶¶ ¶ ¸ ¸¶¶ );

## ���÷¶:

�¶ ¶¶¶¶¶¶¶ʒ¶¶ÿh¶¶¶ ¸ ¶¶¶¶ (efunction)¶¶ ¶т¶Ђ∪¶¶¶¶¶¶ (eoperator) ¶¶т¶¶¶¶¶¶¶¶б¶. ¶¶¶û¶¶¸¶¶¶¶¶¶, ¶¶¶e¶¶¶¶ ¸ ¶¶¶¶¶¶ʒ¶¶ /OPCPROF.efun ¶¶ /OPCPROF.eoper¶¶ ¶¶¶¶¶¶¶¶¸¶¶¶. ¶¶¶¸¶¶¶¶h¶¶¶¸¶¶¶, ¶¶¶¶¸¶¶¶¶¶¶¶¶¶¶¶¶ʒ¶¶¶¸¶¶¶¶¶.

## ¶¶¶o¶:

¶¶ function_profile(3)

## ¶¶¶¶¶¶:

¶¶ Spock @ FF 97.Apr.24.¶¶ (printed 3/16/95)

---

# query_load_average(3) MudOS v21c2�� (5 Sep 1994)

## ������:

	�� query_load_average()�� - ���� mud ��єтѕ������ (load average).

## ���:

	�� string query_load_average( void );

	�� ._��  query_load_average( ( ( 

# refs(3)���h MudOS v21c2h (5 Sep 1994)

---

## ׀׀׀׀׀׀:

׀׀ refs - ׀׀׀׀h׀׀׀׀׀ ű׀IJo׀׀׀׀׀.

## ׀׀׀:

׀׀ int refs( mixed data );

׀׀ ׀׀׀׀ refs( ׀׀׀ data );

## ׀׀׀÷׀:

׀׀ refs() ׀׀׀ data ׀IJo׀׀׀׀׀.
׀׀׀׀׀׀׀׀׀׀׀׀Ĵ׀׀׀ data ׀׀ǰ.
׀ж׀׀p׀Ç׀׀׀׀h׀׀׀׀׀ ű.

## ׀׀׀o׀:

׀׀ children(3), inherit_list(3), deep_inherit_list(3), objects(3)

## ׀׀׀׀׀׀:

׀׀ Spock @ FF 97.May.24.׀׀ (printed 3/16/95)

---



---

# rusage(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� rusage() - ������єт���� (system call) getrusage() ѻ������Ц.

## ���:

```
�� mapping rusage( void );

�� ў�� rusage( ѻ   ��ѻ );
```

## ���-ѻ:

```
�� ����  ���ѻ (efun)ᵋ�ѻ getrusage()
ѻ єт������ѻ���Ц. ������ getrusage()
ѻ�����ѻ˷ѻ��Ρ-ѻ̌ѻ-ѻ������Ц����.
ѻe-єтû-ѻ getrusage(), ������ times() ѻ-єт����.
ѻ������єт-ѻ-ѻ utime ѻ stime ѻ-ѻ-ѻ-ѻ-ѻ.
ѻ-ѻ-ѻ-ѻʰ-ѻ艷λѻ  ѻ (millisecond).

ѻ ѻ-ѻ-ѻ-ѻ rusage() -ѻķѻ-ѻ:
ѻ
```

```
    ѻ void
    ѻ create()
    ѻ {
        ѻ-ѻ-ѻ mapping info;

        ѻ-ѻ-ѻ info = rusage();
        ѻ-ѻ-ѻ write("'ѻ-ѻ-ѻʰѻ = " + info["utime"] + "  ѻ
        \n");
        ѻ-ѻ-ѻ write("єтʰѻ = " + info["stime"] + "  ѻ\n");

    ѻ }
ѻ ѻ-ѻ-ѻ'ѻõѻ-ѻL-ѻ:
ѻ
    ѻ utime, stime, maxrss, ixrss, idrss, isrss, minflt, majflt, nswap,
```

inblock, oublock, msgsnd, msgrcv, nsignals, nvcsw, nivcsw.

## ����o�:

•• time_expression(3), function_profile(3), time(3), uptime(3)

## ������:

•• Spock @ FF 97.Jun.2.•• (printed 3/16/95)

---

# set_debug_level(3) MudOS v21c2�� (5 Sep 1994)

## ������:

�� set_debug_level() - ��趨������ (driver) ''�� debug() ��
(macro) ħ, ����� (debug) ĵ¢.

## ���:

```
�� void set_debug_level( int level );

��  ���̡set_debug_level( ���  level );
```

## ���÷:

```
�� ����  ���  (emulated function, efun)
ֵ���ڣ�������������ħ''�  -DDEBUG_MACRO
ֵ����Ч. set_debug_macro() ��  mud��  ��ħ,
ֵ���Ղ��ڣ����Ƴ�����Ш�������������.

�  ��o������������'  ���e  debug.h,
ֵ�������þ��Ķ���.
```

## ���o:


�  set_malloc_mask(3)


## ������:


��  Spock @ FF 97.Jul.20.��  (printed 3/16/95)

# set_malloc_mask(3) MudOS v21c2�� (5 Sep 1994)

## ������:

�� set_malloc_mask() - ��趨 ������گ����õij�������������’������ (the mask controlling display of malloc debug info).

## ���:

�� void set_malloc_mask( int mask );

�� ��  ���ˌset_malloc_mask( ���� mask );

## ���÷�:

�� ����  ���� (efun)ˌ ���گ��ˌ mud�� ������ (driver) ʱ, options.h ˌ DEBUGMALLOC�ˌ DEBUGMALLOC_EXTENSIONS ˌˌʱ����ʱ����Ч. ˌˌ (mask) ˌˌˌˌˌˌˌˌˌˌˌˌˌˌˌˌˌˌˌˌگ��иˌˌگ�ʱ, ˌˌ’ˌˌˌˌˌˌˌˌص���گij����Ш. ˌˌþˌˌˌˌˌˌˌˌˌˌˌˌˌˌˌˌˌˌˌˌ’ ˌˌˌeˌ md.c

## ���o:

ˌˌ debugmalloc(3)

## ������:

ˌˌ Spock @ FF 97.Jul.21.ˌˌ (printed 3/16/95)

# swap(3)���� MudOS v21c2�� (5 Sep 1994)

## ������:

�� swap -¸ ï¿½h¿½¿½¿½¿½¿½ (swap) ¿½z.

## ����:

�� void swap( object );

�� ¿½ ¿½¿½¸swap( ¿½¿½ );

## ¿½¿½¿½:


¿½¿½ ¿½¿½¿½ ¿½¿½¿½¿½¿½

# time_expression(3) MudOS v21c2�� (5 Sep 1994)

## ������:

�� time_expression() - ��� ���h��'�� (expression) ��������ıñ��.

## ���:

�� int time_expression( mixed expr );

�� ���� time_expression( ��� expr );

## ���÷�:

�� ���� expr,�� ���� ��(microsecond)�Ï��λ��� ��� expr ������ñ��. ��������l'ȷ�Ÿ�����Ç��ȷ��h ��, t����, ����ą̈l'ȷ�õ ��.

## ���o�:

�� rusage(3), function_profile(3), time(3)

## ������:

�� Spock @ FF 97.Jul.25.�� (printed 3/16/95)

# trace(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

	�� trace() - ��趨¯�  (trace flags) ����  ɵı¯.

## ���:

	�� int trace( int traceflags );

	�� ���  trace( ���  traceflags );

## ���-�:

	�� ��趨¯�  
	  ������ɵ˝��  ����  .¯  ħ, ��ʒ  ���Ш.

	�� ������������Ш (trace bits):

		�  Trace all function calls to lfuns.
		��  (������жЪ¸�������í)

		�  Trace all calls to "call_other".
		��  (������ж call_other ĵ)

		�  Trace all function returns.
		��  (������к����Ļش)

		�  Print arguments at function calls and return values.
		��  (ʒ������ú����IJ,
		)

		�  Print all executed stack machine instructions
		��  (produces a lot of output!)
		��  (ʒ�������¸eĶv
		(  !) )
		
		�  Enable trace in heart beat functions.

[illegible]:

[illegible] traceprefix(3)

[illegible]:

[illegible] Spock @ FF 97.Jul.25.[illegible] (printed 3/16/95)

---



---

# traceprefix(3)�� MudOS v21c2�� (5 Sep 1994)

---

## ������:



## ���:

	�� string traceprefix( string prefix );

	�� ��_���� traceprefix( ��_���� prefix );

## ���÷�:

	�� ������趨��ǰ�j��ǰ (traceprefix) (������ï 0),
	ˏ����������������ǰj������ˍ�������������Ž������.

## ���o:

	�� trace(3)

## ������:

	�� Spock @ FF 97.Jul.25.�� (printed 3/16/95)

---



---

# ? ???? ???? ű??

## ??????????ek????ļ????




ص???hx

# allocate_mapping(3) MudOS v21c2�? (5 Sep 1994)

---

## ??????:

?? allocate_mapping() - ?????? (pre-allocate) h???ǰ??h??ŷ?? (mapping).

## ???:

?? mapping allocate_mapping( int size );

?? ŷ?? allocate_mapping( ???? size );

## ???-?:

?? ????h???????? size ??  ?? (elements) ??ŷ??.

## ?? ????:

```
???? mapping x;
???? int y = 200;

???? x = allocate_mapping(y);
```

?? y ?????ŷ??ij?'??C. ??????ĜĎ???ŷ???ж??, '?? allocate_mapping ?DZŁMõķ??? (?200 ???????Ĵ ??  ??) , ??Ïh?ξ??趨??Ç???õ??,?ǰ ǰ?????'?????DZŁ???Ч?ı?. ?????Ç??ŷ?廩 ?? 200 ???????, ?? allocate_mapping ?æ??"ɔ???Ч?ı̣?'.

?? ʊ?? !?? ?????Ɋ?2?ŗ??ŷ???e??????  ??, ??Ö???? x = ([]) ????'?????????e?ŷ?? x, ????ñ??? allocate_mapping().

•• گ• :••ש•, +••••ÿ••Ĵ•C••û•••••••.

•• x = allocate_mapping( 200 ); •••• x = ([]);

••••o•:

•• map_delete(3)

••••••:

•• Spock @ FF 96.Oct.14.•• (printed 3/16/95)

---

•ص•••hx

---

# each(3)���� MudOS v21c2�� (5 Sep 1994)

---

## ������:

�� each() - ?η???h??ў?????? (mapping) ?к??? (key, value) ??????.

## ???:

�� mixed *each( mapping m, int reset );

## ???-?:

�� each()?? ???????h??ў??????????ў?? (key, value) h??h?ε?????. each() ????????Ĵ?????? keys() ?? values() ??????????????Ĵ???h??. ?? each() ????ў????????β??ħ, ???h???ȷ?????? (null vector, ?? (0,0)). ???  ??Ĵ?????ŏ??, ??h?? each() ???ص??????ˏ??и??'.

?? ??????????'?? each() ??ą????ў?????ħ, h????????и??'. ??Ƈ MudOS ?e?ˌ???? (single thread) ???????, ????Χ?????????? each() ??ħ?????????. (ɻ????h???????  ???'?? each() ??dў?言???????h??ħ, ????Lў?? (key, value) ??λ??)

??
????????Ĵ????????ĭ
1, ?? each() ??????´?и??'??ą.

## ?? ????:

```
?? mixed *pair;
?? while ((pair = each(x)) != ({})) {
```

```
        ???? write("key???? = " + pair[0] + "\n");
        ???? write("value = " + pair[1] + "\n");
        ?? }
```

???o?:

?? keys(3), values(3)

??????:

?? Spock @ FF 97.Feb.7.?? (printed 3/16/95)

---

?ص???hx

---

# filter_mapping(3)���h MudOS v21c2������ (5 Sep 1994)

## ������:

�h filter_mapping() - ��h������Ï,
ŸY�h��ў������e¤ijЦ ¤ ¤ ¤

# keys(3)���� MudOS v21c2�� (5 Sep 1994)

## ������:

	�� keys() - ������h��ÿ����� (mapping)
	������ (�  ���, ����) (�� (key, value) )
	���  ��¨����� (array).

## ���:

	�� mixed *keys( mapping m );

	�� ��� *keys( ÿ�� m );

## ���:

	�� keys()�� ��  ̊ ̊ ̊ h����ÿ����� m�� ��
	(̊ ̊ ̊ 

# map_delete(3)�� MudOS v21c2�� (5 Sep 1994)

---

## ������:

### ��

map_delete() -��
�ƌ- ���h��ў�������r��h�� (�  ���,
����) ((key, value)).

## ���:

### ��

void map_delete( mapping m, mixed element );

### ��

map_delete( ў�� m, ���
element );

## ���:-�:

### ��

map_delete r��ў������й  ��_��� element�� ��
(�  ���, ����) .

### ��

### ��

```
mapping names;

names = ([]);
names["truilkan"] = "john";
names["wayfarer"] = "erik";
names["jacques"] = "dwayne";
```

### ��

### ��

```
map_delete(names,"truilkan");
```

�� ��'ў����� names ��Ї:

### ��

```
(["wayfarer" : "erik", "jacques" : "dwayne"])
```

�� map_delete( names, "truilkan" ) ��, key(names)
Ґ������ truilkan (���Å������ ("truilkan", *)

•• truilkan •ţ•ŷ••••••••)

••••o•:

•• allocate_mapping(3)

•••••••:

•• Spock @ FF 97.Feb.18.•• (printed 3/16/95)

---

•ص•••hx

---

# map_mapping(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� map_mapping() - ����h������ ̈h��ŷ������e ̈ ̈ ̈

# mapp(3)���� MudOS v21c2�� ( 5 Sep 1994 )

---

## ������ :

�� mapp() - ж h ı p Ïў (mapping).

## ��� :

```
int mapp( mixed arg );
```

```
mapp( ��� arg );
```

## ���-� :

arg Ñ ў , 1.

## ���o :

stringp(3), pointerp(3), objectp(3), intp(3), bufferp(3), floatp(3) functionp(3), nullp(3), undefinedp(3), errorp(3)

## ������ :

Spock @ FF 97.Mar.14.�� (printed 3/16/95)

---

# match_path(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� match_path() - ��h��ў����� (mapping) ����Ψ·������ (path)

## ���:

�� mixed match_path( mapping m, string str );

�� ��� match_path( ў�� m, ¸��� str );

## ���÷�:


�� match_path() ��h��ў�������·����. match_path() ������eĹ ��� (key) ï¸��� (string) ����. ������¸ (value) ��œ��  ¸Ç. ����  ���� (efun) ���χ������������  ¸���. ��  / ��β¸Ĺ ��¸˛χ¸  ¸¸˛e·������, ¸ / ����h˛¸¸¸¸¸ (wildcard), ��ï¸·¸¸¸¸µ¸κζ¸¸ (¸: ¸ ¸¸¸ļ¸¸·¸¸¸¸¸õ¸ * ¸¸).


## ������:


�� Cygnus


## ������:


�� Spock @ FF 97.Apr.13.�� (printed 3/16/95)

# unique_mapping(3) MudOS v21c2�� (5 Sep 1994)

## ������:

�� unique_mapping() - ��h������Ï,
��h������h��ў�� (mapping).

## ����:

```
�� mapping unique_mapping( mixed *arr, string fun, object ob, mixed
extra, ... );
�� mapping unique_mapping( mixed *arr, function f, mixed extra, ... );

�� ў�� unique_mapping( ��� *arr, ��̣��  fun, ��  ob,
��  extra, ... );
�� ў�� unique_mapping( ��  *arr, ������̣  f, ��
extra, ... );
```

## ���÷�:

```
�  ������µķ'������h��ў��:

��  ����������̣ij�Ư (member) ����h��,
���Ŀ�̣̣̣Ï  ��  (key).
```

## ���o:

```
��  filter_array(3), sort_array(3), map(3)
```

## ������:

```
��  Spock @ FF 97.jul.25.��  (printed 3/16/95)
```

# values(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� values() - ��h��ÿ������ (� ���, ����) ( (keys, values) )�� ��, ������鶹������.

## ���:

�� mixed *values( mapping m );

�� ��� *values( ÿ�� m );

## ���-�:

�� values() ����h������. ������ÏÿÏÿ����m�� ��, (� ���, ����) ������������ ��.

### ��:

```
mapping m;
m = (["hp" : 35, "sp" : 42, "mass" : 100]);
```

### ��:

```
values(m) == ({35, 42, 100})
```

�� ��ص��, ��ш������, ��  ��。����  ��。��h��.

## ���o:

�� keys(3), each(3)

## ������:

�� Spock @ FF 97.jul.25.� (printed 3/16/95)

# ? ????? ????? mud ??????

---

## ??????????eķ????ļ???



---


ص???hx


---

# author_stats(3)�� MudOS v21c2�� (5 Sep 1994)

---

## ������:

	�� author_stats() - ������  hЩЧ������.
	�� domain_stats() - ��������hЩЧ������.

## ���:

	�� mapping author_stats( string domain | void );
	�� mapping domain_stats( string domain | void );

	�� ў�� author_stats( ̲��  domain ̲ void );
	�� ў�� domain_stats( ̲��  domain ̲ void );

## ���÷:

	�� author_stats() ̲ domain_stats()̲
	̲̲̲h̲̲̲̲̲ў̲̲̲̲̲̲̲̲̲̲.
	̲̲̲û̲м̲̲с̲̲̲ (argument) , ̲э̲̲̲̲̲̲̲̲̲
	(̲̲̲̲̲̲̲̲̲̲) ̲ў̲̲̲̲ (̲̲ў̲̲̲̲̲̲)
	̲̲ў̲̲̲̲̲̲̲. ̲ў̲̲ў̲̲̲̲̲  ̲̲̲̲̲: ̲ƶ̲̲̲
	(moves), ̲ķ̲ʰ̲̲̲̲ (cost), ̲̲̲̲̲̲ (errors),̲̲
	̲̲̲̲̲̲ (heart_beats), ̲̲̲̲ (worth), ̲̲̲̲̲̲
	(array_size),̲̲ ̲̲ ̲̲̲̲̲ (objects).
	ўh̲  ̲̲j̲̲̲̲ص̲̲J̲̲ў̲̲̲̲̲̲̲e̲̲̲̲̲.
	̲ƶ̲̲̲̲̲̲̲̲̲̲

# ([ domain0 : info0, domain1 : info1, ... ])

([ "moves" : moves, "cost" : cost, "errors" : errors, "heart_beats" : heart_beats, "worth" : worth, "array_size" : array_size, "objects" : objects ])

## SEE ALSO

domain_file(4), author_file(4), set_author(3)

## AUTHOR

Spock @ FF[illegible] 96.Oct.15.[illegible] (printed 3/16/95)

---

# domain_stats(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

	�� domain_stats() - ����������hЩЧ�������.
	�� author_stats() - ��������  hЩЧ�������.

## ���:

	�� mapping domain_stats( string domain | void );
	�� mapping author_stats( string domain | void );

	�� ў�� domain_stats( ̣���  domain ��  void );
	�� ў�� author_stats( ̣���  domain ��  void );

## ���÷�:

	�� author_stats() ��  domain_stats()��
	������h�������ў�������������.
	���û�м��c���  (argument) , ̣э�������������
	(̣̣̣̣̣̣̣̣̣̣) ÿ����  (̣̣ÿ������)
	̣̣ў������. ỵ̣̈ў������  ������: ̣ƶ���
	(moves), ̣ķ̣ʰ����  (cost), ������  (errors),̣̣
	������  (heart_beats), ̣̣̣̣  (worth), ̣̣̣̣̣̣
	(array_size),̣̣  ̣̣  ̣̣̣̣̣  (objects).
	ÿḥ  ̣̣ǰ̣̣̣̣ص̣̣Ĵ̣̣ў̣̣̣̣̣̣̣ẹ̣̣̣̣̣.
	̣ƶ̣̣̣̣̣̣̣̣̣̣

# ([ domain0 : info0, domain1 : info1, ... ])

([ "moves" : moves, "cost" : cost, "errors" : errors, "heart_beats" : heart_beats, "worth" : worth, "array_size" : array_size, "objects" : objects ])

## SEE ALSO

domain_file(4), author_file(4), set_author(3)

## AUTHOR

Spock @ FF[illegible] 96.Oct.15.[illegible] (printed 3/16/95)

---

# enable_wizard(3) MudOS v21c2�� (5 Sep 1994)

## ������:

�� enable_wizard() - ����ъ����Ę����h�����.

## ���:

�� void enable_wizard( void );

�� ��  ���̡ enable_wizard( ¤  ��¤ );

## ���÷�:

�� ¤кє¤¤¤ enable_wizard() ¤Ьц¤¤¤ (interactive) ¤¤¤, ¤¤¤¤ wizardp() ¤Ъ¤¤¤¤¤¤¤¤¤¤¤̡ (true). ¤¤ enable_wizard() ¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤Ę:
¤¤

1. 1.¤¤ ¤¤¤¤¤¤¤¤ MUD ¤¤¤¤¤¤¤¤ʰ¤¤¤¤ RESTRICTED_ED w¤¤, ¤¤¤¤¤′¤¤ ed ¤¤¤¤¤¤¤¤ġ‘ (restricted modes).
1. 2.¤¤ ¤¤¤¤¤ȷ¤¤¤þ¤¤̣д¤¤¤¤¤Ш.
1. 3.¤¤ ¤¤¤¤′¤¤ trace(3) ¤¤ traceprefix(3) ¤¤¤¤.

## ¤¤¤o¤:

¤¤ disable_wizard(3), wizardp(3)

## ������:

¤¤ Spock @ FF 97.Feb.12.¤¤ (printed 3/16/95)

# export_uid(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� export_uid() -�� ��趨 h������'����ʁ������ (user id, uid).

## ���÷:

```
�� int export_uid( object ob );

�� ���� export_uid( ��( ob );
```

## ���÷:

�� ��趨 ob ��'����ʁ������ (uid) Ï this_object() ����Ч'����ʁ������ (effective user id, effective uid). ������`���� ob ����Ч'����ʁ������Ï 0 ƒ��Ч.

## ���o:


�� this_object(3), seteuid(3), getuid(3), geteuid(3), previous_object(3), valid_seteuid(4)


## ������:


�� Spock @ FF 97.Feb.12.�� (printed 3/16/95)

# find_living(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� find_living() - ��x��h������ķ������ (id) ��Ļ (living)�� ���

## ���:

�� object find_living( string str );

�� ��� find_living( ������ str );

## ����-�:

�� ������ų��Ï���ŵ (living) , ������ķ������ (id) Ï str ������. ��Ĝ�����ù enable_commands()�� ������ű���Ï���ŵ����. ���Ţ������������ set_living_name() ��趯����, ��h���ŪŻ��Ž���Ψ�� (hash table) ��������Ψ�h������ ��.

## ����o:

�� living(3), livings(3), find_player(3), users(3), enable_commands(3) set_living_name(3)

## ������:

�� Spock @ FF 97.Feb.12.�� (printed 3/16/95)

# geteuid(3) MudOS v21c2 (5 Sep 1994) geteuid(3)

## ??????:

?? geteuid() - ????h??????????h??????????Ч'????к? (effective user id, euid)

## ???:

?? string geteuid( object | function );

?? ?_??? geteuid( ??? ?? ?????_?? );

## ???÷?:

?? ??????????h?????, geteuid ??????????Ч'????к?????? (euid) . ????_???IJ?????h?????_?? (function pointer,?? ????????Ї function), ???ش?????????_??????? euid . ????????????_??ı???, ?گ?????????ʰû?? euid,?? ???'?????????'????к?????? (user id, uid) ????.

## ???o?:

?? seteuid(3) getuid(3) functionp(3) export_uid(3) previous_object(3) this_object(3) valid_seteuid(4) ??

## ??????:

?? Spock @ FF 97.Feb.14.?? (printed 3/16/95)

# getuid(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� getuid() - ����h������'����«������ (user id, uid)

## ���:

```
�� string getuid( object ob );

�� ?��? getuid( ??? ob );
```

## ???:

```
�� ????h?????'????«?????.
h????????? uid?? ?????????fi ?? creator_file()
?????????.
```

## ???o?:

```
?? seteuid(3), geteuid(3), export_uid(3), this_object(3),
previous_object(3), creator_file(4), valid_seteuid(4)
```

## ??????:

```
?? Spock @ FF 97.Feb.14.?? (printed 3/16/95)
```

# living(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� living() - ж h ̣ p ššš

## ���:

```
int living( object ob );
```

```
living( ob );
```

## ���-:

ob h ŵ (true, ).  (X ~, ob ù enable_commands() )

## ���o:

interactive(3), enable_commands(3)

## ������:

Spock @ FF 97.Feb.18. (printed 3/16/95)

---

# livings(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� livings() - ��������л��� (living)
�����h�������

## ���:

```
�� object *livings( void );

�� ��� *livings( ?   ??? );
```

## ���-?:

```
�? ?????????л????????? (pointer)??
??h????????.
?? ???w?????????????????ù?
enable_commands()?? ?????.
```

## ???o?:

?? enable_commands(3), find_living(3), users(3), objects(3)

## ������:

?? Spock @ FF 97.Feb.18.?? (printed 3/16/95)

# query_privs(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� query_privs() - ����h������_���_��� (privs string).

## ���:

�� string query_privs( object ob );

�� ��_��� query_privs( ��� ob );

## ���÷�:

�� ����h������_���_���.
�� ������������ĕ��ĕ (master object)
ĕ۞ĕ������ħ, privs_file()�� ��������.
������_������ set_privs()�� ��������.

�� query_privs() ������������ (driver) ����ħ,
���ĕ PRIVS ����Ч.

## ���o�:

�� privs_file(4), set_privs(3)

## ������:

�� Spock @ FF 97.Jul.25.� (printed 3/16/95)

# set_author(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� set_author - ��趨��h������ص�����.

## ���:

�� void set_author( string author );

�� ��  ���̲set_author( ̲���author );

## ���÷:

�� Ï��ÇŦ����  ����(domain) ��″��, ���e̲̲̲̲̲̲̲̲̲̲  ̲̲̲̲ص̲̲̲̲̲.
̲̲̲̲̲̲̲̲̲̲

## ����o�:


•• author_file(4), domain_file(4), author_stats(3), domain_stats(3)


## ������:

•• Spock @ FF 97.Jul.20.•• (printed 3/16/95)

---


ص���hx


---

# set_light(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� set_light() - ���Ļ��ğh������������������ (light level).

## ���:

�� int set_light( int light_level_adjustment );

�� ���� set_light( ���� light_level_adjustment );

## ���÷:

�� ⁄⁄ light_level_adjustment ⁄⁄Ï 0,
⁄⁄⁄ğ⁄⁄⁄Ľǰ⁄⁄⁄⁄⁄⁄⁄⁄⁄.
⁄⁄Ï⁄⁄⁄⁄⁄⁄⁄⁄⁄⁄⁄⁄⁄⁄⁄⁄⁄⁄⁄⁄⁄ij⁄,
⁄⁄⁄⁄⁄⁄⁄⁄⁄⁄⁄⁄.

## ������:

�� Spock @ FF 97.Jul.21.�� (printed 3/16/95)

# set_living_name(3) MudOS v21c2�� (5 Sep 1994)

---

## ������:

�� set_living_name() - ��趨h������ŵ (living)
������������������ (living name).

## ���:

�� void set_living_name( string name );

�� ��  ��¸set_living_name( ¸¸¸¸ name
);

## ���÷��:

�� ��h������ŵ������趨������������. ��趨
��, ��������J������ find_living() ��ɳ.

## ����o:

�� enable_commands(3), find_living(3), find_player(3)

## ������:

�� Spock @ FF 97.Jul.21.�� (printed 3/16/95)

---



---

# set_privs(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� set_privs() - ��趨h������_���_��� (privs string).

## ���:

�� void set_privs( object ob, string privs );

��  ���_set_privs( ��� ob, ��_���
privs );

## ���-�:

�� ��趨 privs Ï ob ��_�����.

�� ����  ���� (efun)
_������������������h���� PRIVS ����Ч.

## ���o:

�� privs_file(4), query_privs(3)

## ������:

�� Spock @ FF 97.Jul.21.� (printed 3/16/95)

# seteuid(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� seteuid() - ��h������������Ч'����к������
(effective user id, euid)

## ���:

�� int seteuid( string str );

�� ���� seteuid( ��ֻ str );

## ���ֻ:

�� ��h������Ч'����к������Ï str�� ������. ��
master.c ��e valid_seteuid()
������������h������Ч'����к�����]
趨Ï��.

�� ��h������Ч'����к������Ï 0,
������������'����к������ (uid)��ֻ ����
export_uid() ����.

��ת, ������Ñ����к����.

## ���o:

�� export_uid(3), getuid(3), geteuid(3), this_object(3), valid_seteuid(4)

## ������:

�� Spock @ FF 97.Jul.20.�� (printed 3/16/95)

# wizardp(3)�� MudOS v21c2�� (5 Sep 1994)

---

## ������:

�� wizardp() - ж׳ר׳ף׳ת׳ר׳ף׳ת׳ר׳ף׳ת

# ? ???? ???? ???? ????

---

## ??????????ęķ????ļ???

intp.3
random.3
to_float.3

---

ص???hx

---

# intp(3)���� MudOS v21c2�� (5 Sep 1994)

---

## ������:



## ���:

	�� int intp( mixed arg );

	�� ���� intp( ��� arg );

## ���-�:

	�� ��� arg ��h ѕ  1, ѕ  0.

## ����o:

	�� mapp(3), stringp(3), pointerp(3), objectp(3), bufferp(3), floatp(3), functionp(3), nullp(3), undefinedp(3), errorp(3)

## ������:

	�� Spock @ FF 97.Feb.14.�� (printed 3/16/95)

---



---

# random(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� random() - ����h�������ЪЪ����� (pseudo-random number).

## ���:

�� int random( int n );

�� ���� random( ���� n );

## ���-�:

�� ����h����X�� 0�� ��  n - 1�� ę�������.

## ������:

�� Spock @ FF 97.May.2.�� (printed 3/16/95)

## ���:

�� ������������������ɒ���h������ (seed number), ڪ��ɲ������ɖ��h������, ����Ш������ 0 ��  1ȇ ��.

# to_float(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� to_float - ��h������ת����h�������.

## ���:

�� float to_float( int i );

�� ������ to_float( ���� i );

## ���-�:

�� to_float() ����h�������������, ���, ������ڪ������ i.

## ���o�:

�� to_int(3), read_buffer(3), sprintf(3), sscanf(3)

## ������:

�� Spock @ FF 97.Jul.25.� (printed 3/16/95)

# ? ???? ????? ???

## ??????????ek????į???

# all_inventory(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� all_inventory() - ����h������������������ʒ.

## ���:

�� object *all_inventory( object ob );

�� ��� *all_inventory( ��� ob );

## ���÷�:

�� ����h�������, ������ï��� ob ��������ʒ.

## ���o:

�� first_inventory(3), next_inventory(3), deep_inventory(3)

## ������:

�� Spock @ FF 96.Oct.12.�� (printed 3/16/95)

# children(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� children() - ����h��_������ (object) ������Y_____________.

## ���:

```
�� object * children( string name );

�� ��� *children( _��� name );
```

## ���-�:

```
�� ____________h__________ (array),
���A_____ļ_____Ï name
__________Y_____.
h��'õķ_______x____д'________ (user object)�� ���Y_________:
��
    ���� object *list;

    ���� list = children("/obj/user");
```

```
�� ____________x____e'____ (_____GD____ '____),�� ��
users() ,__ _____ (interactive) '____.
```

## ���o:

```
�� deep_inherit_list(3), inherit_list(3), objects(3)
```

## ������:

```
�� Spock @ FF 97.Jan.26.�� (printed 3/16/95)
```

# clone_object(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� clone_object() - ����h������к������

## ���:

�� object clone_object( string name );

�� ��� clone_object( ̲��� name );

## ���÷�:

```
�� ���  ����
name ����h���µ���� (object) ,
���ч������h�����e�����.
̀'������h���µ����. h������
environment()�� ����̲�����ï 0��
Ļ,
���  ������.
```

## ���o:


�� destruct(3), move_object(3), new(3)


## ������:


�� Spock @ FF 97.Jan.27.�� (printed 3/16/95)

## ������:

�� clonep() -
������h������p�Ï���Z����.

## ���:

�� int clonep( void | mixed arg );
��

�� ���� clonep( ?  ??? | ??? arg );

## ???÷?:

?? ????????? arg ???趨 objectp() ?? O_CLONE ??¯ (flag), ?¯?????и??ѕ??根 (true) ??.(Χ????? 1). MUD ?????????? (driver) ??Ö???? new(3) (clone_object(3)) ??????????????, ?趨 O_CLONE?? ??¯. clonep() ?????????????? (master copy,?? ???? call_other(3) ????) ????ѕ??根. ??ш??, ??? clonep() ???ѕ??根, ?? file_name()?? ?Ղ?????????ص??ļ????2???? #n ( n Ï????). clonep() ?????c?????Ղ??????Ï this_object().

## ???o?:

?? virtualp(3), userp(3), wizardp(3), objectp(3), new(3), clone_object(3), call_other(3), file_name(3)

## ??????:

?? Spock @ FF 97.Jan.27.?? (printed 3/16/95)

# deep_inventory(3) MudOS v21c2�� (5 Sep 1994)

## ������:

deep_inventory() - ����h������������������ʒ,
�����ʒ�л��Ъ������ʒ,
�����Ч�"�ķ�'������ʐ.

## ���:

object *deep_inventory( object ob );

��� *deep_inventory( ��� ob );

## ���÷:

ob ����ʺ������ʒ������ (array),
������ʒχ�����Ψ����ʐ.

## ���o:

first_inventory(3), next_inventory(3), all_inventory(3)

## ������:

Spock @ FF 97.Feb.04.�� (printed 3/16/95)

---

# destruct(3)�� MudOS v21c2�� (5 Sep 1994)

---

## ������:



## ���:

	�� void destruct( object ob );

	��  ���¸destruct( ��� ob );

## ���÷�:

	�� ��õ  ����  ob. ĴǴ destruct()ʺ ��,
	������e�õ¤��  (global variables),
	ɖ¾������Ť���. ��  ob ��  this_object() ,
	ˋ����Þ¸��, ������h����¸.

## ���o:

	��  clone_object(3)  new(3), destruct_env_of(4), move(4)

## ������:

	��  Spock @ FF 97.Feb.04.��  (printed 3/16/95)

---



---

# environment(3)�� MudOS v21c2�� (5 Sep 1994)

---

## ������:



## ����:

	�� object environment( object ob );

	�� ��� environment( ��� ob );

## ���÷�:

	�� ����ę������
	ob ĻĻ������. ����û��ę������ (argument), ob Ļ��Ï
	this_object().

## ���o:

	�� this_object(3)

## ������:

	�� Spock @ FF 97.Feb.12.�� (printed 3/16/95)

---


ص���hx


---

# file_name(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:



## ���:

	�� string file_name( object ob );

	�� ̣���� file_name( ��� ob );

## ���÷:

	�� file_name() ������ ob ʰ���õ��ļ�����.
	������Ñ��Y�����, ��  file_name()��
	���ص����Ü�����ı���Θŵ������õ�������
	������  ������ļ�������̃����ę#��  ��,
	ڀ������ĸ��ŏ. ����Þ��ʰ,
	����ĸ��ŏ�Ŵ 0��  ��’����,
	ÿ����h�������̈́̈́̈́̈́̈́̈́̈́̈́̈́̈́h��.
	����ÿh�����Ÿ́̈́̈́̈́̈́̈́̈́̈́̈́̈́̈́̈́L����ص
	���ụ̂̊̈́̈́̈́ ob, ��  ob Ɽ̨��Ï this_object().

## ������:

	�� Spock @ FF 97.Feb.12.��  (printed 3/16/95)

# find_object(3)�� MudOS v21c2�� (5 Sep 1994)

---

## ������:

�� find_object() - ��ˌ������ļ�����Ψ��h�����.

## ���:

�� object find_object( string str );

�� ��� find_object( ˌ��� str );

## ���÷:

�� ���ļ����� str�� Ψ��h�����.

# first_inventory(3)���� MudOS v21c2������ (5 Sep 1994)

## ������:

�� first_inventory() - ����h������ن��j̑h����ʒ

## ���:

```
object first_inventory( mixed ob );
```

```
first_inventory( ���ob );
```

## ���÷:

```
ob ������ ob ن��j̑h����ʒ, ob
h������������������ļ�����.
```

## ���o:

file_name(3), next_inventory(3), all_inventory(3), deep_inventory(3)

## ������:

Spock @ FF 97.Feb.12. (printed 3/16/95)

---

ص���hx

# load_object(3)�� MudOS v21c2�� (5 Sep 1994)

---

## ������:



���ą�������Ψ�h�����h�����


## ���:



## ���÷�:



����ą����Ĵ�������δ����,



## ���о:



## ������:



---


ص���hx


---

# master(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� master() - ������������ (master object)

## ���:

```
�� object master( void );
```

```
�� ��� master( ?   ??? );
```

## ???-?:

�� ?????????????? (master object) ??? (pointer).

## ????:


�� find_object(3)


## ??????:


�� Spock @ FF 97.Apr.13.�� (printed 3/16/95)

# move_object(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� move_object() - ��Ľǰ������ҩ�����Ļ���(environment) ��.

## ���:

�� void move_object( object item, mixed dest );

�� ̈ ̈ ̈ move_object( ��� item, ��� dest );

## ���÷:

�� ����� item ҩ��� dest ����. item ������ this_object(). ��� item ̈ this_object(), move_object() ̈̈̈̈̈ dest ḧ̈̈̈̈̈.

## ���o:

�� this_object(3), move(4)

## ������:

�� Spock @ FF 97.Apr.15.̈ (printed 3/16/95)

# new(3)���� MudOS v21c2�� (5 Sep 1994)

## ������:

	�� new() - ���  ����h������.

## ���:

	�� object new( string name );

	�� ��� new( ̲��� name );

## ���÷�:

	�� ̲ name
	������ļ�������������h���µ����,
	���ч������h������ص������.
	ˆ̀̀̀̀̀̀̀̀̀̀̀̀̀̀̀µ����. h̲̲ environment()
	Ï̲̲̲

# next_inventory(3) MudOS v21c2�� (5 Sep 1994)

## ������:

�� next_inventory() - ����h������������ (inventory)�e���h�����.

## ���:

�� object next_inventory( object ob );

�� ��� next_inventory( ��� ob );

## ���-�:

�� ������������������������, ob ˜�����h�����.

�� ����: ������ ob ⁄⁄ move_object()⁄⁄ ����, ⁄⁄ next_inventory() ����µ�������б⁄⁄з������.

## ���o�:

⁄⁄ first_inventory(3), all_inventory(3), deep_inventory(3)

## ������:

⁄⁄ Spock @ FF 97.Apr.24.⁄⁄ (printed 3/16/95)

---

# objectp(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� objectp() - ж h ı р Ïh (object).

## ���:

```
int objectp( mixed arg );
```

```
objectp( ��� arg );
```

## ���-�:

arg h и 1.

## ���o�:

mapp(3), stringp(3), pointerp(3), intp(3), bufferp(3), floatp(3), functionp(3), nullp(3), undefinedp(3), errorp(3)

## ������:

Spock @ FF 97.Apr.24.�� (printed 3/16/95)

# objects(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� objects - ����h������������̃������������ (array).

## ���:

�� object *objects( string func, object ob );
�� object *objects( function f );

�� ��� *objects( ���� func, ��� ob );
�� ��� *objects( ������ f );

## ���÷�:

�� objects() ���� mud��
���Ğ̃�������̃������������. ��ʊ��, ���єт鄆
��������鱠�� (maximum array size)_��, objects()
���  ����������ص��ж������ŋĵ  ط.
����h��, objects() X_��û��íôỗ.

�� ������ func ��  ob ����������, ������ ob->func(), ������������Ğ̃������������Ï func() ЮIJ���. ������������ط���¸ ������ɔ objects()�� ������. ��������  ̃.

�� ����_ f,
������������������������ú����_ f.
����: objects( (: clonep :) )
̃  ����_õ���к������������б̃.

## ���õ:

�� livings(3), users(3)

## ������:

# present(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� present() - ����Ʒ�ı"���� (id) Ψ�����.

## ���:

�� object present( mixed str, object ob );

�� ��� present( ��� str, ��� ob );

## ���-�:

�� ���h�������������Ï str, ѝ��ش����.

�� str Ҳ�������h�����, ������ŁƆ�����.

�� ����ٟ ob, present() ��Ψ�� ob ������ (inventory), ��Ƈ��Ψ��Ľǰ, ѣ�Ľǰ.

## ���o�:

�� move_object(3), environment(3)

## ������:

�� Spock @ FF 97.Apr.24.�� (printed 3/16/95)

# query_heart_beat(3) MudOS v21c2�� (5 Sep 1994)

## ������:

�� query_heart_beat() - ��ğh������������ (heart beat) ״��.

## ���:

�� int query_heart_beat( object );

�� ���� query_heart_beat( ��� );

## ���÷�:

�� ������Ğ̃Ɔ̃ù̃ set_heart_beat() ������, query_heart_beat() ������.
������ ob, ��Ɍ̨Ɍ̃ÏĿǰ̃.
û̃̃̃

# reload_object(3) MudOS v21c2�� (5 Sep 1994)

## ������:

�� reload_object - ��h������ ɸn������ˮ.

## ���:

�� void reload_object( object ob );

�� ˙ ˙ ˙ reload_object( ��˙ ob );

## ���÷�:

˙˙ ˙˙ ob ˙˙˙˙ reload_object() ʰ,
˙˙˙˙

# restore_object(3) MudOS v21c2�� (5 Sep 1994)

## ������:

�� restore_object() -
��h���į�����h������ı���.

## ���:

�� int restore_object( string name, int flag );

�� ���� restore_object( ̣��� name, ���� flag );

## ���÷:

�� ̨ɣ��� name ���į�����Ľǰ����ı���. ��� flag ̀ 1, ̀Ọ́̀̀̀̀ (non-static)
̀̀̀̀̀J̀̀̀ǰ̀̀̀̀̀ï̀̀. (̀̀″̀̀̀̀,
̀Ø̣̀̀̀̀̀

# save_object(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� save_object() - ��h������ı����_���泛 h���ļ���.

## ���:

�� int save_object( string name, int flag );

�� ���� save_object( ?_��? name, ???? flag );

## ���-�:

�� ������������ekØ?_????? (non-static variables) ???泛????? name ???ļ???.
?????????Ķ?????????????? (master object) ?e? valid_write() ?????????p?????????????д??.
?????J?????????_??Ï 1,
?????Ï??ĵı???X?z???z
(h??~??????????).
????????? ,ı?????z??Ï 0.

## ???????:

�� save_object() ?ı???? 1, ʧ и??? 0.

## ????o?:

�� restore_object(3)

## ??????:

�� Spock @ FF 97.Jul.4.?? (printed 3/16/95)

# set_heart_beat(3) MudOS v21c2�� (5 Sep 1994)

## ������:

�� set_heart_beat() - ��'��ą��h������������ (heartbeat).

## ���:

�� int set_heart_beat( int flag );

�� ���� set_heart_beat( ���� flag );

## ���÷�:

�� ������Ï 0�� ą������������.
�� ������Ï 1��
��'������ÿh������������ heart_beat()
����h��.��
(ÿh�������������ˇ������������, ����
mud�� ĺ����  ��Ƨ��岖������)
����Ï���Ï���Ï 1�� ������,
�������ÿ����ʿ������ŵ���h��
heart_beat(). ��Çʊ������, ���Ï mud��
������ ĺ€ᴛ趨Ï����Ï 1 ������Ï 1.

## ���o:

�� heart_beat(4), query_heart_beat(3)

## ������:

�� Spock @ FF 97.Jul.21.�Ï (printed 3/16/95)

# set_hide(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� set_hide() - ��h������Ը�������趨�����-(hide flag).

## ���:

```
�� void set_hide( int flag );
```

```
�� ��  ���_set_hide( ���� flag );
```

## ���÷:

```
�� ������ flag ��
h������IJ����¯�� .¯
1 ��  ��0 ����,
������ɋ���J'����〃、���  master()->valid_hide(ob)
Ղ����屁����ˍ,                    ���. ������崦
〃˘���ڳ,
、'''''''
```

# tell_object(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� tell_object() - ����Щ���  h�����.

## ���:

�� void tell_object( object ob, string str );

�� ��  ���_tell_object( ��� ob, ��_���
str );

## ���-�:

�� ����Щ str�� ���  ���� ob. ��� ob
��h���ı����� (interactive) ��� (���),
�„���Щ�И ob ��'. ��, ��л  ob ǐ���
catch_tell().

## ���o�:

�� message(3), write(3), shout(3), say(3), tell_room(3)

## ������:

�� Spock @ FF 97.jul.24.�� (printed 3/16/95)

# tell_room(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

tell_room - ��h������e�������Т���Ш.

## ���:

```
void tell_room( mixed ob, string str, object *exclude );
```

```
   ___tell_room( ___ ob,
____ str, ___ *exclude );
```

## ���÷�:

```
____Ш str__ ___  _____ ob
e_________. ob _____Ґ_____ў_____.
_____________________ exclude, __ exclude
____________________ц____ѫ___Ш.
```

## ���o:

message(3), write(3), shout(3), say(3), tell_object(3)

## ������:

Spock @ FF 97.Jul.24.�� (printed 3/16/95)

# virtualp(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� virtualp() - ж̣����ı����p̣��h������ (virtual) ���.

## ���:

�� int virtualp( object arg );

�� ���� virtualp( ��� arg );

## ���÷:

�� ����� arg�� ��ˇ, objectp() Ï 1, ���趨 O_VIRTUAL ��¯, virtualp()�� ̈ӥ̈̈

# ? ???? ???? ? ???

---

## ??????????ek?????l???



---


?ص???hx


---

# parse_command(3)�� MudOS v21c2���(5 Sep 1994)

## ����ɳ

�� parse_command() - ����Ł�����‘(pattern)����h����

## ��  ��

```
�� int parse_command( string command, object env|object *oblist, string
pattern, mixed arg, ... );

�� ��  parse_command( ��  command, ��  env|
��  *oblist, ��  pattern, ��  arg, ... );
```

## ���-���

```
�� parse_command() ������������sscanf(3)
ġ������h���‘��h��Ŀ
sscanf(3)����Ղ������IJ(refence)
sscanf(3)���×parse_command()
IJ
```

```
�  ' command' ��Ղ  parse_command()  1
```

```
 'env'  'oblist'
h  ɔ h  
ġh Ļ  Ч
deep_inventory ‘ Ɔh  養
ġ Ч ġ
```

```
    parse_command(cmd, environment(), pattern, arg)
```

```
      ���� parse_command( cmd, ({ environment() }) +
      deep_inventory(environment()), pattern, arg)
�� ������h����_������ " 'get' / 'take' %i "
�� ��  ��
      ����  'word'��������������������������
      ڔ������(text)
      ����  [word]������������������������
      ��?Ъ�����
      ����  /
      ������������������������������������
      w���ЪĮÁ(marker)
      ����
      %o������������������������������
      ��h��Ł(item), ���
      ����
      %l������������������������������
      ����(living object)
      ����
      %s������������������������������
      ҡ�����
      ����
      %w������������������������������
      ҡκ���(word)
      ����
      %p������������������������������
      ����(list) (��є��)
      ����
      %i������������������������������
      ҡ���Ł
      ����
      %d������������������������������
      ����  0- ����  tx(0-99)
�� 'arg'
���п�������������Ни�������������Щ����
sscanf ҈ЈĽ�������ъ҈ÿh  ������������Çh��
%��

�� ��%_ ҈ķ�������æ҈
```

```
        ����  %o������������  ����h������
        ����  %s������������  ���ص��ـ��ـ���
        ����  %w������������
        ���ص�h��ـ��ـ���
        ����  %p������������
        ������h�����e��ـ����e�������h�
            ������  ���Ĺ
                ��������
                ����ǿǎ�����h���ـ���
                ��������  ��������嶪array[0]
                ������Ъĵ���
        ����  %i������������
        ����h�����и'˙���ض����嶪
            ������ [0] = (int) +(wanted) -(order) 0(all)
            ������ [1..n] (object) ����ـ��(objectpointers)
        ����  %l������������
        ����h�����и'˙���ض����嶪
            ������ [0] = (int) +(wanted) -(order) 0(all)
            ������ [1..n] (object) ����ـ��
                ��������  ��Ш̧��������
            ����  %d������������  ����h����ـ
    ����  õ������������Ш̧��  % ����ـ
    %i ��  %l����ı̧ш̧����  %l
    ڳ̣ڳ(parse)ǰ̧̧̧
```

```
                ������ }
    ��

�� MUDLIB¸   ��

    ��
    Ï��'����  ��������ã������������������
    mudlib ��¸ ������Çh䴀
    ��������жм���(sensible
    manner)�ь������ǰ���ã���ą������Щ��

    ��
    ������ǰ
    汾�у���'���  LPC ������h���  id() lfun
    ���ц�h���ض��̣������̣������������������

    ��  ��µÏ汾������������ş������  'names' ���e
    LPC ������
    ��

        ����  1 - h鯈h������̣
        ����  2 - ������'ǩ̀̀̀
```

# process_string(3) MudOS v21c2�� (5 Sep 1994)

## ������:

�� process_string() - ���ٟ������Ъ��ú����Ľ�����ijЩ��.

## ���:

�� string process_string( string combinestring );

�� ٟ���� process_string( ٟ���� combinestring );

## ���÷�:

�� ������к'���h�������e�ijЩ��. ������'�  ĴĴ��ú�������ص��ٟ����.

�� ��'����:

�� "@@������[:ę]�[ | ���1 | ���2....| ���N ]@@"

��
������п��Ъ�������eí������

�� ęٟ[]->[]( ���1, ���2, ....., ���N );

�� ��ш��, process_string() ����  (recuse) ���÷��صٟ����ժ����ٟ.
[]ٟ[]صٟ[]ٟ[], process_string() ���Ш���h�η��ص[]ٟ[].

�� ��combinestring[]
e[]ٟ[]ٟ[]ٟ[]í[], []ٟ[]ٟ[], ����Ĵ��ø'����Щٟ[]á[]ٟ[]ط[]ٟ[]

# process_value(3) MudOS v21c2�� (5 Sep 1994)

---

## ������:

�� process_value() -
��ą́Í������ą̨ú����ķ́,

## ���:

�� mixed process_value( string calldescription );

�� ��́ process_value( ą́́ calldescription );

## ���ą́:

�� ��ą́к‘ą̨ú����ķ́, ‘‘́:

```
���́ "��������[:ę́ķ́]][ | ���́ 1 |
���́ 2.... | ���́ N]"
```

�� ����ą́́ќ,

�� ��ш,
������ę́ķ́о́Ш’́пп  [] ����ſ́.
������Щ́ХД́ą́́.

## ���о́:

�� process_string(3)

## ������:

�� ́ mud́ ́ (mudlib) ,
́’ú́õķ́, ‘‘́ process_string() ǰ, ‘‘ ‘‘́ (effuserid, euid)́ Ї 0́ .

# query_verb(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� query_verb() - ��������.e�������.

## ���:

�� string query_verb( void );

�� ..... query_verb( ...... );

## ...:


�� .....Ljj.e.....
.....û.....к.....,  .  0.
.....;..:
...ü.....h.....h,
.жє.....h.....ô¯...


## ...o.:


�� add_action(3)


## ������:


�� Spock @ FF 97.May.2.�� (printed 3/16/95)

# ? ???? ???? ???? ????

---

## ??????????ęķ????ļ???



---


ص???hx


---

# socket_accept(3)�� MudOS v21c2�� (5 Sep 1994)

## ������ц

�� socket_accept() - ����h�� socket ������(connection) ��

## ���  ��

```
�� #include <socket_err.h>

�� int socket_accept( int s, string read_callback, string write_callback );

�� ����  socket_accept( ����  s, ̣����  read_callback,
̣����  write_callback );
```

## ���÷���

```
�� ����  s ��h������  socket_create(3)
̣̣̣̣̣̣̣̣  socket_bind(3)
̣̣̣̣h��λ̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣(listen)��
socket_listen(3)̊  ̣̣̣̣̣ụ̈  socket��socket_accept()
��δ̣̣̣̣̣(pending)̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣wd̨��ḥ̣̣̣̣
s ̣̣̣̣̣Ъ̣̣̣̣  socket
̣̣̣̣̣̣̣̣̣ḥ̣̣µ̣̣̣̣̣̣̈(descriptor)̣̣̣̣̣̣̣̣̣̣
̣̣̣̣̣̣̣̣̣̣̣̣̣̣Þ̣Ĵ̣̣̣̣̈                  ̣  socket
̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣ḥ˝̣д̣̣̣̣
socket s ̣̃  ̣̣̣̣Ћ̣̣̣δ̣̣̣̣̣̣̣̣ʒ̣
```

```
�� read_callback
̣̣̣̣̣̣̣̣̣Ṇ̣̣̣̣̣̣̣̣̣̣̣̣̣̀ḥ̣̣µ̣
sockeṭ̣̣̣̣̣̣̣̣Ç̣          ̣
sockeṭ̣̣̣̣̣̣̣̣̣̣ħ̣̣̣õỊ̣̣̣̣̣̣́ợ̣̣̣̣̣̣̣ḑ
(callback) Ọ̣̣̣̣̣̣̣̈и̣‘̣̣
```

### ???? void read_callback(int fd)

?? ?????? fd ??  ?????????? ? socket??

?? write_callback
??????????Ǹ????????????????µ?
socket?????????Ç ?
socket??????д??ɦ???õÍ???????ơ????д??ė?
??

### ???? void write_callback(int fd)

?? ?????? fd ???????д??? socket??

?? ʊ?⁞?????? socket
????????ر?Σ???
socket_close(3)
?ĵ??ý?????Ļ?????????????Ç ?
socket????????µ? socket????
close_callback??????ر?ɛ???Ö????????¸?'??
??

### ???? void close_callback(int fd)

?? ?????? fd ???oγر? socket??
??

## ?????????

?? socket_accept() ?گı?ʰ???h?????????ƍ???? socket
?ķǸ?????????ʃ??ʰ?????????h????¨??soc
??????????õ?????????????????????輝
??

## ?????????

?? EEFDRANGE??????????
????¨???????X??

?? EEBADF??????????????????
??ч????????¨??

?? EESECURITY????????? ??єY????ō??

•• EEMODENOTSUPP•• ••¸ •• socket ••••

•• EENOTLISTN••••••••• socket
•••••••••C•

•• EEWOULDBLOCK•••• ••••••••••••(block)••

•• EEINTR••••••••••••••••••
•ж€•єт•••á•

•• EEACCEPT••••••••••••• ••••••••:·

•• EENOSOCKS••••••••••• û•ж•••• efun
sockets••
••

•••••q•

•• socket_bind(3), socket_connect(3), socket_create(3), socket_listen(3)
••

•••••

•• Kenny@Broken.History•• 97.Jul.26•• (printed 3/16/95)

---



---

# socket_acquire(3)�� MudOS v21c2�� (5 Sep 1994)

## ������ɳ

### ��  socket_acquire() - ����h��  socket ������Ę(ownership)

## ���  ��

```
#include <socket_err.h>

int socket_acquire( int socket, string read_callback, string
write_callback, string close_callback );

socket_acquire( ����  socket, ę����  read_callback, ę����  write_callback, ę����  close_callback );
```

## ���÷���

### ��  ����  socket_acquire() ��Ï��Ç��ɲ�’��  socket_release() Çґ��h��  socket ����Ę���J���Ę����h������������є������ę◂ ������Ǹ���ü�������Ę�������иŻȩ(callbac socket Ł���Ę��ʒ���Ǹ���ü�������Ę������������◂ socket������Ր Ļ�����Ç����  socket_acquire() ������������ґ�ơ������Ր�Ļ������������ȩ◂ 践  ��g

### ��  ��ǰh������  £������̈́������ɲ������������Ǹ  socket ��ü���  ���Ш��ģ��д��  ر2ȩ������Ҳ  MudOS ������̈́̈́̈́

•• socket_acquire()
••••••••иŻگ••••••ع••••ã•••ِ•••••Ù•
socket••

••••••ą•

•• socket_release(3)

•••••



•• Kenny@Broken.History•• 97.Jul.27•••• (printed 3/16/95)


---


•ص•••hx


---

# socket_address(3)�� MudOS v21c2�� (5 Sep 1994)

## ������ɳ

�� socket_address() - ����h�� efun socket ��Ω��λ

## ���  ��

```
#include <socket_err.h>

string socket_address( int s );
```

## ��ˍ���

socket_address() ���� efun socket s ��Ω��λ�����ص�λ��'���£

"127.0.0.1 23".

## ��������

socket_address() ���Ǐ

```
    ɹ��ʰ��h
    ɹ��ʰ��h
```

## ������ɻ

socket_connect(3), socket_create(3), resolve(3), query_host_name(3), query_ip_name(3), query_ip_number(3)

## ����

Kenny@Broken.History�� 97.Jul.27�� (printed 3/16/95)

# socket_bind(3)�� MudOS v21c2�� (5 Sep 1994)

## ������ц

�� socket_bind() - ����(bind)h�� socket

## ���  ��

```
�� #include <socket_err.h>

�� int socket_bind( int s, int port );

�� ���� socket_bind( ���� s, ���� port );
```

## ���-���

```
�� socket_bind() Ïh��δ������ socket
-��h������ě���h�� socket ��  socket_create(3)
������������V������h�������ɒ酞λ_��,
address family������û��_������ě socket_bind()
��Ç������Ï socket s ������
```

## ��������_��

```
�� socket_bind() ���Ì

    ���  ɾ�ʰ��  EESUCCESS��

    ����  ʃ��ʰ��h������������������ĸ_���
```

## ��������_��

```
�� EEFDRANGE�������������
����_(descriptor)������X��

�� EEBADF�������������������
��Ч������_��
```

•• EESECURITY•••••••••• ••cY•••••õ••

•• EEISBOUND•••••••••••• socket ••••••••••

•• EEADDRINUSE••••••• ••λ•ψ•'•á•

•• EEBIND•••••••••••••••••••• ••••••••:.

•• EEGETSOCKNAME•• getsockname •••:.

••••••q•

•• socket_connect(3), socket_create(3), socket_listen(3)

•••••

•• Kenny@Broken.History•• 97.Jul.27•• (printed 3/16/95)

---



---

# socket_close(3)�� MudOS v21c2�� (5 Sep 1994)

## ������ц

�� socket_close() - ��ر�h�� socket

## ���  ��

```
�� #include <socket_err.h>

�� int socket_close( int s );

�� ���� socket_close( ���� s );
```

## ���-���

```
�� socket_close() ��ر socket
s������ицh����'õ socket efun ���(slot)��
```

## ��������

```
�� socket_close() ���ĺ

    ���� ��ѧ֏ EESUCCESS��

    ���� ʃ��֏Å�������������к_���
```

## ��������

```
�� EEFDRANGE������������
    ����(descriptor)������X��

�� EEBADF�������������������
    ��Ч������_��

�� EESECURITY������ ��єY����ō��
```

## ������ц

•• socket_accept(3), socket_create(3)


•••••


•• Kenny@Broken.History•• 97.Jul.27•• (printed 3/16/95)


---


ص•••hx


---

# socket_connect(3)�� MudOS v21c2�� (5 Sep 1994)

## ������ɳ

�� socket_connect() - ��’��h�� socket ϵ������(connection)

## ���  ��

```
�� #include <socket_err.h>

�� int socket_connect( int s, string address, string read_callback, string
write_callback );

�� ����  socket_connect( ����  s, ̲����  address,
̲����  read_callback, ̲����  write_callback );
```

## ���÷���

```
�� ����  s ��h��  socket��s ������h��  STREAM
���ɔ���h��  MUD ���  ̲  socket��address ������
socket ��Ç̲̲̲̲̲̲̲̲ü̲λ̲́λ̲��  "127.0.0.1 23"
̲̲̲̲̲‘̲̲

�� read_callback
̲̲̲̲̲Ǹ̲̲̲̲̲̲̲̲̲̲̲̲̲̲̲̲̲̲̲̲̲̲̲̲h̲ˬõ̲̲̲̲

    ̲̲̲̲  void read_callback(int fd, mixed message)

    ̲̲  ̲̲̲̲̲  fd ̲̲Ç̲̲̲̲̲̲̲̲  ̲  ̲  socket̲̲̲̲
    message ̲æ̲̲ȷ̲̲̲̲̲̲̲  ̲  ̲

    ̲̲  write_callback ̲̲̲̲̲Ǹ̲̲̲̲̲̲̲̲̲̲̲̲̲̲̲̲̲̲̲
```

## ������q


•• socket_accept(3), socket_close(3), socket_create(3)


## ●●●●●


•• Kenny@Broken.History•• 97.Jul.27•• (printed 3/16/95)


---


ص●●●hx


---

# socket_create(3)�� MudOS v21c2����
# (5 Sep 1994)

---

## ������ɳ

�� socket_create() - ����h�� efun socket

## ���  ��

```
�� #include <socket_err.h>

�� int socket_create( int mode, string read_callback, void | string
close_callback );

�� ���� socket_create( ���� mode, ¸���  read_callback,
¸  ��  | ¸���  close_callback );
```

## ���¸���

```
�� socket_create() ����h�  efun socket��mode
������Ç¸������������¸¸¸
```

# ᦀæᦀᦃᦀᦀᦀᦀᦀ ᦀ

᦬᦬ close_callback ᦀᦀᦀᦀᦀ

## ⬜⬜ socket_accept(3), socket_bind(3), socket_close(3), socket_connect(3), socket_listen(3), socket_write(3)


## ⬜⬜⬜⬜⬜


⬜⬜ Kenny@Broken.History⬜⬜ 97.Jul.26⬜⬜⬜⬜⬜⬜ (printed 3/16/95)


---


⬜ص⬜⬜⬜hx


---

# socket_error(3)�� MudOS v21c2���� (5 Sep 1994)

## ������ɳ

socket_error() - ����h socket ������������

## ���  ��

#include <socket_err.h>

string socket_error( int error );

## ���÷���

socket_error() ����h

## ��������

socket_error() ���Ĵ

> ��ɪ�h

> ������ʹ "socket_error: invalid error number"
> ������
> ����ʊ"socket_error: ЧĴ"

## ������ɳ

socket_create(3), socket_connect(3)

## ����

Kenny@Broken.History 97.Jul.27 (printed 3/16/95)

# socket_listen(3)�� MudOS v21c2����
# (5 Sep 1994)

---

## ������ɳ

�� socket_listen() - ����(listen)h�� socket
������(connection)��

## ���  ��

```
�� #include <socket_err.h>

�� int socket_listen( int s, string listen_callback );

�� ���� socket_listen( ���� s, ̣��  listen_callback );
```

## ���÷���

```
�� ï��Ç�������ī�����Ç�� socket_create(3)
����h�� socket������ socket_listen(3) ����  socket
ŵ������ġ'������������ socket_accept(3)
�������������ʒ socket_listen() ������̧��������
STREAM ����  MUD ���  ̧ sockets��

�� listen_callback
������Ñ�������������Ĵõ������������ socket
������ʰ���õí������ơ�������������ɇ(callba


    ���  void listen_callback(int fd)

�� ������ fd ������������e socket��
```

## ̧̧̧̧̧̧̧

```
�� socket_listen() ���Ì
```

```
    ����  ̣ı̣ʰ EESUCCESS
```

## ERRORS

- EEFDRANGE
  (descriptor)
- EEBADF
- EESECURITY
- EEMODENOTSUPP socket
- EENOADDR socket
- EEISCONN socket
- EELISTEN

## SEE ALSO

- socket_accept(3), socket_connect(3), socket_create(3)

## AUTHOR

- Kenny@Broken.History 97.Jul.27 (printed 3/16/95)

# socket_release(3)�� MudOS v21c2���
# (5 Sep 1994)

## ������ɳ

�� socket_release() - ��и�h�� socket ������Ę (ownership) ����h�����

## ���  ��

```
�� #include <socket_err.h>

�� int socket_release( int socket, object ob, string release_callback );

�� ���  socket_release( ����  socket, ���  ob, ę���
release_callback );
```

## ���-���

```
�� socket_release() ��������ı  ę̨̨̨̨̨̨̨̨̨趨
(set-
up)̃̃̃̃̃̃̃̃̃̃Ę̃̃̃J̃̃̃Ę̃̃̃̃̃̃̃ת̃̃h̃̃̃
socket ̃̃̃̃h̃̃̃̃Ø·̃δ̃̃̈́̃á̃

�� socket
̃̃̃̃Ę̃̃̃ת̃ữ̃̃̃̃̃h̃̃̃Ľǰü̃̃̃̃̃̃̃Ę̃̃̃̃̃̃̃
socket
̃̃̃̃̃̃̃̃̃̃̃̃̃̃̃̃̃̃(handshake)̃̃̃̃̃̃̃̃̃̃̃̃̃̃̃̃
socket_release() ̃̃’̃ġ̃socket_release()
̃̃̃̃̃̃̃ő̃Ъ̃̃̃̃̃̃̃̃(integrity)̃̃̃̃̃̃  ̃̃̃̃̃̃
ob ̃̃ release_callback ̃̃̃̃̃̃̃̃̃̃̃̃̃̃̃̃̃̃
```

# socket_write(3)�� MudOS v21c2���� (5 Sep 1994)

## ������ɳ

�� socket_write() -ᵠ��h�� socket ��������

## ���  ��

```
�� #include <socket_err.h>

�� int socket_write( int s, mixed message, void | string address );
```

�� ���� socket_write( ���� s, ��� message, ᵠ  ��� | ᵢ��  address );

## ���÷���

```
�� socket_write() ��h�� socket s
������������  ᵢ���  socket s ᵠ STREAM ᵢ���
MUD ᵢ��ᵠ᷊ᵢ᷊ socket
ᵠᵸ᷊᷊᷊᷊᷊᷊᷊᷊ᵠᵤ᷊(connected)
ᵢ᷊᷊᷊Ⱬᵢ᷊᷊᷊ᵢ᷊ᵢ᷊᷊᷊᷊᷊ socket ᵠᵠ DATAGRAM
ᵢ᷊᷊᷊ᵠᵸᵠ᷊᷊᷊ᵢ᷊ᵢ᷊ᵢ
```

# [illegible]

## [illegible]ęķ[illegible]ļ[illegible]

# break_string(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� break_string() - ��Ω-��ļ����h����_��� (string).

## ����:

�� string break_string( string str | int foobar, int len, void | int indent | string indent );

�� ��_��� break_string( ��_��� str ��  ����  foobar, ����  len, ��  ��  ��  ����  indent ��  ��_��  indent );

## ���÷:

�� break_string() ��ÿ len��  ����_ (character) ���h��û�л��e����_���, ����ÿh����ɛ�T�к��澽��. ��  indent ��h����_���, ��ÿ��T���ت���ĭ������ indent.

�� ������h������ (argument) ��h������, break_string() ��_��  0.

## ������:

�� break_string()��
ĵ������ō����_������������  ..
������������ɛþ��  '.'  ɔ���  '\n' .��
������������eĽ��,
������û������Ɋ�Ĵ�h��. ɳl'�����:
��Ö���ஃ��ĕ���h���h, Ҳ������  '\n' ,
Ȼ����'��  break_string(). ��Ҳ���c����  break_string() ɸş�к'. ������  MudOS ɨ汾��, Ҳ������ break_string()  ɛ  '\n' .

## ���o:

•• implode(3), explode(3), sprintf(3)


•••••••:


•• Cygnus


•••••••:


•• Spock @ FF 96.Oct.16.•• (printed 3/16/95)


---


•ص•••hx


---

# capitalize(3)�� MudOS v21c2�� (5 Sep 1994)

---

## ������:

�� capitalize() - ��h����_����j�h��Й����_(character) п�н�д.

## ���:

```
�� string capitalize( string str );

�� ?_???? capitalize( ?_???? str );
```

## ???-?:

```
�� ?V?????_??? str ?j?h??Й????_?п?н?д,
??????h??п?????????_???.
```

## ???o?:

```
�� lower_case(3)
```

## ??????:

```
�� Spock @ FF 96.Oct.16.?? (printed 3/16/95)
```

---



---

# clear_bit(3)�� MudOS v21c2�� (5 Sep 1994)

---

## ������:

�� clear_bit() - ���h���_����e�ijh�� bit .

## ���:

�� string clear_bit( string str, int n );

�� ��_���( clear_bit( ��_���( str, ���( n );

## ���÷�:

�� ����h���µ��_��( (string) , ������_���e n�� ��( bit ���Ï 0. ��ש��, ������_��( str�� ��û�и�������.

## ���o:

�( set_bit(3), test_bit(3)

## ������:

�( Spock @ FF 97.Jan.27.�( (printed 3/16/95)

---



---

# crypt(3)�? MudOS v21c2�? (5 Sep 1994)

## ������:

�? crypt() - ??h????_????????б???.

## ???:

?? string crypt( string str, string seed );

?? ?_???? crypt( ?_???? str, ?_???? seed );

## ???-?:

?? ?????_??? seed ??j?????_?????????????, ????_???? str?? ???б???. ??? seed Ї 0?? , ?э?????????_??б???.

?? ?????_?????j???????к??Ї???????.

## ??????:

?? Spock @ FF 97.Jan.27.?? (printed 3/16/95)

# explode(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� explode() - ���h����_���.

## ���:

�� string *explode( string str, string del );

�� ��_���* *explode( ��_���* str, ��_���* del );

## ���÷�:

�� explode() ����h����_�������������* (array),
������������_��* str ** del ��_��є�T**.

## ������:

**** explode( str, " " );

**** ��������h** str
******Й**j**_**********(**** str
**Й**ї***, ўh**********ıh**_***).

## ���o**:

**** sscanf(3), extract(3), replace_string(3), strsrch(3)

## ������:

**** Spock @ FF 97.Feb.12.** (printed 3/16/95)

# implode(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� implode() - ��������_���.

## ���:

�� string implode( mixed *arr, string del );

�� ��_��� implode( ��� *arr, ��_��� del );

## ���÷�:

�� ������ arr
п����ц�������������������h��,
ÿh����_���ˇ�������_��e
������в��_����  ��(element)
�����ժ���.

## ���o:


�� explode(3) sprintf(3)


## ������:


�� Spock @ FF 97.Feb.14.�� (printed 3/16/95)

# lower_case(3)�� MudOS v21c2�� (5 Sep 1994)

---

## ������:



## ���:

	�� string lower_case( string str );

	�� ¸���¸ lower_case( ¸���¸ str );

## ���÷�:

	�� ¸¸¸¸h¸¸¸¸¸¸¸¸¸¸Сд��'
	( ¸¸¸¸¸¸_¸¸¸¸¸¸¸¸¸¸¸¸) .

## ���o:

	�� capitalize(3)

## ������:

	�� Spock @ FF 97.Feb.18.�� (printed 3/16/95)

---



---

# reg_assoc(3)�� MudOS v21c2���� (26 Dec 1994)

---

## ������ǫ

### ��

reg_assoc() - h�������‘ (regular pattern) ����ˍ���  ę��(extractor)

## ���  ��

### ��

mixed *reg_assoc(string str, string *pat_arr, mixed *tok_arr, void | mixed def);

### ��

*reg_assoc(ˍ��  str, ˍ��  *pat_arr, ���  *tok_arr, ˍ  ���  |  ���  def);

## ���ˉ-���

### ��

reg_assoc ��h���ˍ�����ȥ(explode)ÏhIIIĮ���þ��  pat_arr ˍ����’(regular expression)��‘ˍ�������ˍ�������Ih  ˍ����  tok_arr ˍ����  ˍ  ��  def��Ґ  0ˍ����ˍ��Ļ˙˙˙˙˙˙˙˙˙h��δ˙˙˙˙˙˙˙˙˙˙  養��h����h˙˙˙˙ˏ˙‘˙˙˙˙˙˙

#### ˍ˙˙˙ ({ δ˙˙˙ 1, ˙˙˙ 1, δ˙˙˙ 2, ˙˙˙ 2, ..., δ˙˙˙ n, ˙˙˙ n, δ˙˙˙ n+1 })

### ��

˙˙˙ Ĵ˙˙˙˙˙˙˙˙˙˙˙˙˙Ö˙˙˙˙˙˙  ˙˙˙˙˙˙˙

#### ˍ˙˙˙ ({ def, ˙˙˙˙˙˙˙˙ 1 ˙˙˙˙˙˙˙˙, ..., def, ˙˙˙˙˙˙˙˙ n ˙˙˙˙˙˙˙˙, def })  ��

### ��

pat_arr ˙˙ tok_arr ˍ˙˙˙h˙˙˙˙C˙˙tok_arr ˙ĵ˙ i ˙˙  ˙˙˙Hu˙Ö˙˙ pat_arr ˙ĵ˙ i ˙˙  ˙ƽ˙pat_arr ˙˙˙˙ˏ˙˙˙˙˙ˍ˙˙˙˙

`�` `��` pat_arr`��`
tok_arr`��`Ĵ`C`Ļ`` ({ ({ str }), ({ def }) })`

`ī`

`#define STRING_PAT "\"(\\\\.|[^\\\"])*\""`
`#define NUM_PAT "[0-9]+"`

`#define F_STRING 1`
`#define F_NUM 2`

`reg_assoc("Blah \"blah\" test 203 hhh j 308 \"bacdcd\b\"acb",`

`({ STRING_PAT, NUM_PAT }), ({ F_STRING, F_NUM }), "no-match")`

`` `` ``

`` ({ ({ "Blah ", "\"blah\"", " test ", "203", " hhh j ", "308", " ", `
`` "\"bacdcd\b\"", "acb" }),
`` ({ "no-match", F_STRING, "no-match", F_NUM, "no-match", F_NUM,
`` "no-match", F_STRING, "no-match" }) })

`` ``

`` Kenny@Broken.History`` 97.Jul.26`` (printed 3/16/95)

---



---

# regexp(3)�� MudOS v21c2���� (5 Sep 1994)

## ������ц

regexp() - ��������‘(regular expression)�������

## ���  ��

string *regexp( string *lines, string pattern, void | int flag );

string *regexp( string *lines, string pattern, void | int flag );

## ���÷���

h����Ђ����Ƣ������������������������
h������Æ��з��ҭ‘
願��flag��Ղ��0����λ 2 ���趨
���  ��ј������  ��ƿ����  ������ flag
��λ 1 ���趨‘ ({ ��ʊ
ʊ ,... ,1 ���  ,1 + 1 n + 1,  n
}) ������養�1 ��ʊ
h�ҭ��h�ѵ��Dz

## ������‘  ��

h ‘Hu `|'
ķ(branch)кη ЩҬ h

h(piece)өġ

h h‘Iu `*'`+' `?'
ĵ(atom) h
`*'ĵ h h Hu єĵ h `+'

`xabbby' ��ں��`ab*' ��������
`abbbb'��ע  ����Ķ `xabyabbbz'
��ں�������������Ъ `x' ���
`ab'��������ā������������  ’����ģ�h��Ç

## ����o���


•• sscanf(3), explode(3), strsrch(3), ed(3)


## ·····


•• Kenny@Broken.History•• 97.Jul.26•••• (printed 3/16/95)


---


ص���hx


---

# replace_string(3) MudOS v21c2�� (5 Sep 1994)

---

## ������:

�� replace_string() - ��
h������̲̲̲̲з̲̲̲̲̲̲̲̲̲̲̲̲̲̲.

## ���:

```
�� string replace_string( str, pattern, replace );
�� string replace_string( str, pattern, replace, max );
�� string replace_string( str, pattern, replace, first, last );

�� string str, pattern replace;
�� int max, first, last;

�� ̲̲̲̲ replace_string( str, pattern, replace );
�� ̲̲̲̲ replace_string( str, pattern, replace, max );
�� ̲̲̲̲ replace_string( str, pattern, replace, first, last );

�� ̲̲̲̲ str, pattern replace;
�� ̲̲̲̲ max, first, last;
```

## ���-�:

```
�� replace_string() ̲̲ str ̲̲̲̲̲з̲̲̲̲̲̲̲̲̲̲̲̲̲̲
pattern ̲̲̲̲̲̲̲̲̲̲̲̲̲̲
```

### ����������ŭIJ����������µĹ���:

�� ���к������ first:

��ˌ ��h������̨̊̈́̀̀̀

# set_bit(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� set_bit() - ��趨h��λ ̲���(bitstring) ��e��h��λ (bit).

## ���:

```
�� string set_bit( string str, int n );

�� ̲��(  ̲��(  str, ����  n );
```

## ���-�:

```
�� ����h���µ��̲���, ���̲������ str ��eĵ n
λ ��趨Ï 1. ��ʊ��,  ̲̲̲̲̲̲̲̲ str
��û̀ѝ̀̀̀̀̀̀.

�� n ̲̲̲̲̲̲̲̲̲̲̲̲̲̲̲̲.
̲̲̲̲̲̲̲̲̲̲̲̲̲̲̲̲
```

# sprintf(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� printf, sprintf - ת������ַ����ʽ������.

## ���:

```
�� void printf( string format, ... );
�� string sprintf( string format, ... );
```

```
�� ��  ���ַ printf( ַ���� format, ... );
�� ַ���  sprintf( ַ���  format, ... );
```

## ���÷�:

```
�� ����Ï LPC (s)printf() ��к�ʽ÷ֵ, Lynscar (Sean A Reith)
������hЩ�����÷ֵ.

�� Ŀǰִ֧  ���µ��  к�ʽ (modifier):

�� " " ������ַַ֧����h���n֮.

�� "+" ������ַַ֧����h��  +֮  ֮.

�� "-" ������λ (field)֮  ֵ.

    ֵ  ֮  ֮ :֮ʹ֮
    (s)printf()֮
    ֶ֮֮Hʊ֮֮֮ɣ֮,
    ֮֮Hʊ֮֮Çַ֮֮֮֮֮Ï֮֮֮
```

•••••••:

•• sprintf() ••••_••••'••••_•••.

•••••••:

•• Sean A. Reith (Lynscar)

•••о•:

•• sscanf(3)

•••••••:

•• Spock @ FF 97.Jul.23.•• (printed 3/16/95)

---



---

# sscanf(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� sscanf() - ��h���_����ж�����_����'��������.

## ���:

�� int sscanf( string str, string fmt, mixed var1, mixed var2 ... );

�� ���� sscanf( _��� str, _��� fmt, ��� var1, ��� var2 ... );

## ���÷�:

�� ��_����к' fmt�� ç�����_��� str. fmt ®r��� %d ��%s ����, ��ц_____Ї��'. ÿh�� %d ��%s ����Öh_____, ��_°��Ї var1, var2 ... %d�� ����h_____, ��%s ����h���_���. * _____Ц'��, (��%*d�� ��%*s) ����Á (*)�� ��'_____ ______. (Ҳ���Dz��V____ _____)�� LPC ��sscanf() ��C ��sscanf() ��Ć____, _____в̃��. LPC ��sscanf() ____Ç (Ҳ____) ė_____ڑ̀λ_̀ sscanf(),�� ̀Ç_____. ̀̀h_____, LPC ̀̀ sscanf() ____: sscanf( str, "%s %s", str1, str2 ); ���, _____ str ®ej̀h̀̀Й̀ĵ̀̀̆ (____h___h̀̀_̀̀ǰ_____) ____ str1, _____ ___ str2.

̀̀ sscanf() ̀̀ᒐ̀̀þ'̀̀ %d ̀̀ %s ____.

## ���o:

̀̀ explode(3) replace_string(3) strsrch(3)

# strcmp(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� strcmp() - ж������_��������Є (lexical relationship).

## ���:

�� int strcmp( string one, string two );

�� ���  strcmp( _��  one, _��  two );

## ���-�:

�� ������ strcmp() ��  C��  ������ strcmp ��. ����_��  one ������ two��ĭ (��  one�� ��Й����e_ ���  twõ ĭ),��  ��  strcmp()  ����ر 0��  C��_:  ������������, ¤  ¤ 0.  ��  two��  ���_ ����  one��ĭ, ��  strcmp()��  ����ر 0��  ���_:  ____  ____  (efun)  ��'��  sort_array(3)��  ____ħ, ر_____.

## ���o�:

�� sort_array(3)

## ������:

�� Spock @ FF 97.Jul.23.�� (printed 3/16/95)

# stringp(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� stringp() - ж h p ï

(string).

## ���:

�� int stringp( mixed arg );

�� ���� stringp( ��� arg );

## ���-�:

�� ��� arg h

## ����o:


�� mapp(3), pointerp(3), objectp(3), intp(3), bufferp(3), floatp(3), functionp(3), nullp(3), undefinedp(3), errorp(3)


## ������:


�� Spock @ FF 97.Jul.23.



(printed 3/16/95)

# strlen(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� strlen() - ����h����_�����ij���.

## ���:

�� int strlen( string str );

�� ���� strlen( ?_???? str );

## ���-?:

�� strlen() ????h????_?????????ж?? ??_? (characters).

## ????o?:

?? sizeof(3)

## ������:

?? Spock @ FF 97.Jul.23.?? (printed 3/16/95)

# strsrch(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� strsrch() - ��h����_������Ψ���ص����_���.

## ���:

�� int strsrch( string str, string substr | int char, int flag );

�� ���� strsrch( _���� str, _���� substr ��  ����
char, ���� flag );

## ���-�:

�� strsrch() ����_��  str��  ��Ψ�ц�h������_
substr. ���  flag Ï -1,��  ��Ψ�����h������_ substr.
����J������������������, ��_Ψ��h���_
(character) (��r C��  ��  strchr() ��  strrchr() ). strsrch()
�  ¯Ψ�ę��_������ (empty string) ó_ (null value).

## ������:

�� strsrch()
���ط��ϵ̂h�������h������������ str
e̲λ̲. ̲̲̲û̲з̲̲ϵ̲̲_̲̲̲, ̲  ̲ -1,̲̲
̲̲̲Ǵ̲̲̲̲Ш (̲̲̲̲̲̲IJ̲̲̲̲ɳ̲).

## ���o:

�� explode(3), sscanf(3), replace_string(3), regexp(3)

## ������:

�� Spock @ FF 97.Jul.23.�� (printed 3/16/95)

# test_bit(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� test_bit() - ���h��λ  ̲���� (bitstring) ��ijh��λ (bit)�� ��.

## ���:

�� int test_bit( string str, int n );

�� ���� test_bit( ̲���� str, ���� n );

## ���-�:

�� ������̲��� str ��, ��  n�� ��λ  ̲ 0�� ��  1.

## ������:

�� set_bit(3), clear_bit(3)

## ������:

�� Spock @ FF 97.Jul.24.� (printed 3/16/95)

# 外部函式 —— 系统

以下是现有的翻译文件：

all_previous_objects.3
call_out_info.3
ctime.3
deep_inherit_list.3
error.3
errorp.3
eval_cost.3
find_call_out.3
function_exists.3
function_profile.3
inherit_list.3
inherits.3
localtime.3
max_eval_cost.3
reclaim_objects.3
replace_program.3
reset_eval_cost.3
set_eval_limit.3
set_reset.3
shutdown.3
time.3
uptime.3

回到上一页

# uptime(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� uptime() - ������h������������ (driver) ������������.

## ���:

�� int uptime( void );

�� ���� uptime( ( ) ( ) ( )

# all_previous_objects(3) MudOS v21c2 (5 Sep 1994)

名称:

all_previous_objects() - 传回所有呼叫目前函式的物件阵列.

语法:

```
object *all_previous_objects();
```

用法:

传回一个所有呼叫目前函式的物件阵列 (an array of object). 注意, 区域函式 (local function) 的呼叫不会更动目前的物件 (current object) previous_object() 的内容.

参考:

call_other(3), call_out(3), origin(3), previous_object(3)

翻译:

Spock @ FF 96.Oct.12. (printed 3/16/95)

回到上一页

# call_out_info(3) MudOS v21c2�? (5 Sep 1994)

---

## ??????:

?? call_out_info() - d?????y???e? call_out() ??Ш.

## ???:

?? mixed *call_out_info( void );

## ???-?:

?? d?????????'?? call_out() ????ö??ў??eĺ??????????Ш. ????h???????, ўh??????к? ?? (element): ??? (object), ???? (function), ?ў?h?? (the delay to go), w??IJ??? (the optional argument).

## ???o?:

?? call_out(3) remove_call_out(3)

## ??????:

?? Spock @ FF 96.Oct.16.?? (printed 3/16/95)

---

# ctime(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� ctime() - ����h��ʰ����_���

## ���:

�� string ctime( int clock );

�� ��_��� ctime( ���� clock );

## ���-�:

�� ����h�����ں̄ʰ����_���, ������ clock ���Ղ 1970 ��������������.

## ����o:

�� time(3), time_expression(3), localtime(3), uptime(3)

## ������:

�� Spock @ FF 97.Jan.27.� (printed 3/16/95)

# deep_inherit_list(3) MudOS v21c2�� (5 Sep 1994)

---

## ������:

�� deep_inherit_list() - q��h������������ек����. (parent object)

## ���:

�� string *deep_inherit_list( object obj );

�� ����* *deep_inherit_list( ��* obj );

## ���:

��
����h�������������ж������ы�����,
�����ьХ����Хек����.

## ���o:

�� inherit_list(3) inherits(3)

## ������:

�� Spock @ FF 97.Feb.04.� (printed 3/16/95)

---



---

## ������:

	�� error - ����h��ˌ��ʰ�� (run_time) ����.

## ���:

	�� void error( string err );

	�� ��  ���ˌ error( ˌ��ˌ err );

## ���-�:

	�� ���ˌ error() ʰ, ˌ�����h��ˌ��ʰ�δ��ˌ err.
	����Ľǰ˙˙˙˙˙e˙˙˙˙˙˙˙˙˙ (thread).
	˙˙˙˙˙˙˙˙˙˙ij˙˙˙ ˢ˙ ˙˙˙ij˙˙˙,
	˙˙˙˙¼˙گ˙˙˙˙¼ (debug log) ˙˙˙ˌ˙˙˙.

## ˙˙˙o˙:

	˙˙ catch(3), throw(3), error_handler(4)

## ˙˙˙˙˙˙:

	˙˙ Spock @ FF 97.Feb.12. (printed 3/16/95)

## ������:

�� errorp() - ��ж�h��ٟ���ı����p��ï������ (error code).

## ���:

�� int errorp( mixed arg );

�� ���� errorp( ��� arg );

## ���-�:

�� ������ arg�� ��h������ٟ_ĵ�ĵ, errorp() ���� 1 .

�� ��  ��  ����
(emulated functions, efun),
    ���h_____ĵ_____.
���Ч_____д_____:

```
if (errorp(result = efun_call()))
    printf("error = %d\n", result);
```

�� ���ơ��ļ2���, ���å____h___µí___
perror(result)
_____ĵ_____h_____

## ���o:

�� nullp(3), undefinedp(3)

## ������:

�� Spock @ FF 97.Feb.12.�� (printed 3/16/95)

# eval_cost(3)  MudOS v21c2  (5 Sep 1994)

名称:

eval_cost() - 传回执行耗费时间 (evaluation cost) 还剩多少.

语法:

void eval_cost()

无传回值 eval_cost()

用法:

eval_cost() 在驱动程式 (driver) 判断此时是否陷入一个无穷回圈之前, 传回是否可以执行的指示数字.

参考:

catch(3), error(3), throw(3), error_handler(4), set_eval_limit(3), reset_eval_cost()

翻译:

Spock @ FF 97.Feb.12.  (printed 3/16/95)

回到上一页

# find_call_out MudOS v21c2 (5 Sep 1994)

名称:

find_call_out() - 找到行程中下一个会被呼叫的点召 (call out)

语法:

int find_call_out( string func ):

整数 find_call_out( 字串 func );

用法:

因为执行函式 func 而找到第一个点召 (call out), 并传回剩下的时间. 如果无法找到则传回 -1.

参考:

call_out(3), remove_call_out(3), set_heart_beat(3)

翻译:

Spock @ FF 97.Feb.12. (printed 3/16/95)

回到上一页

# function_exists(3) MudOS v21c2�� (5 Sep 1994)

## ������:

�� function_exists() - ��x������ę̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣ļ̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣.

## ���:

�� string function_exists( string str, object ob );

�� ę̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣ function_exists( ę̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣ str, ��̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣ ob );

## ���÷̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣:

��
ض̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣к̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣
stṛ̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣ ob ļ̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣. str
DẒ̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣ ob
ẹ̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣,
ص̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣ļ̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣p ̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣ file_name(
ob ) ص̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣IJ̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣.

�� ̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣ ob ̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣ụ̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣̂ж̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣庸̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣ stṛ̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣ ̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣ 0.

̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣ ̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣, function_exists() ̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣ ob ̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣Иў̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣ (shadows).

## ���ọ̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣:

̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣ call_other(3), call_out(3), functionp(3), valid_shadow(4)

## ̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣:

̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣ Spock @ FF 97.Feb.12.̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣̣ (printed 3/16/95)

# function_profile(3) MudOS v21c2�� (5 Sep 1994)

## ������:

�� function_profile() -�� ��h������ą�ú����IJ�д (profiling) ��Щ.

## ���:

```
�� mixed *function_profile( object ob );

�� ��� *function_profile( ��� ob );
```

## ���÷�:

```
�� ���ض
ob ���� this_object() (��������ٶ�� ob )
�Í������д��Щ. ������������̨������������
(driver) ����ʱ�ж��� PROFILE_FUNCTIONS
����Ч.
```

## ������:

```
�� ˋ������h��ÿ������������,
ÿh���� ob ėÍ������ص�ÿ���ˋ����:
��
    ��  ([������ "name"��������  :
    ������������,
        ��������  "calls"������  : ���õĴ���,
    ��
    ��  /* cpu ʱ����  ��  (microsecond) Ï��λ */
    ������  "self"��������  : cpu_time_spent_in self,
    ������  "children" : cpu_time_spent_in_children
    ����  ])
    ��  ˋ�����;��  CPU ʱ���й̂, ��Ȼ��λÏ  ��
    (microseconds), ��ɬ���ėď┘Ç��ö̂.
```

## ����o�:


•• rusage(3), time_expression(3), opcprof(3)


## ������:

•• Spock @ FF 97.Jul.26.•• (printed 3/16/95)

---


ص���hx


---

# inherit_list(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� inherit_list() - d̨��h�������̲ek���� (parents) ��嶷.

## ���:

�� string *inherit_list( object obj );

�� ̲��̲ *inherit_list( ��̲ obj );

## ���÷:




## ���o:

deep_inherit_list(3), inherits(3)

## ������:

# inherits(3)  MudOS v21c2  (5 Sep 1994)

名称:

inherits() - 判断一个物件是否继承一个指定的档案

语法:

int inherits( string file, object obj );

整数 inherits( 字串 file, 物件 obj );

用法:

如果 obj 不是继承 file, inherits() 传回 0. 如果 obj 继承最新复制的 file 则传回 1, 如果继承旧的复制的 file 则传回 2.

参考:

deep_inherit_list(3) inherit_list(3)

翻译:

Spock @ FF 97.Feb.14.  (printed 3/16/95)

回到上一页

# localtime(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� localtime() - п���ɱ���ћ��

## ���:

```
�� #include <localtime.h>

�� mixed *localtime( int time );

�� ��� *localtime( ���� time );
```

## ���÷:

```
�� localtime() п��h������������ћ��¸(���� time()
����) ��Ïh������.
�������������Lǰ�ı���ћ��. ����ǰ time()
Á�������ḑ�ø�������ћ�� (GMT (UTC)),
������ћ��Ķ�������������ћ��IJ¸.
������������ķ����Ğ̃Ð̃û̃б̃Ç̃̃. localtime()
������������ʒ̃C̃ћ̃̃̃̃ų̃̃̃̃
```

�� int LT_MIN�� ���� (0..59)
�� int LT_HOUR�� Cʰ (0..23)
�� int LT_MDAY�� ���ۀ (1..31)
�� int LT_MON�� ���ۀ (0..11)
�� int LT_YEAR�� ��ۀ (ۀ������ 1900)
�� int LT_WDAY�� ��ۀ6..0) ۀن)
�� int LT_YDAY�� h��ۀeĵ365..0) ۀۀن)
�� int LT_GMTOFF��
ۀۀۀۀۀۀۀۀʰۀۀۀۀۀۀۀۀۀ (UTC)
�� string LT_ZONE�� ۀۀۀۀʰۀۀۀۀۀۀۀ

ۀۀۀoۀ:

�� ctime(3), time(3), time_expression(3), uptime(3), /include/localtime.h

ۀۀۀۀۀۀ:

�� Spock @ FF 97.Feb.18.�� (printed 3/16/95)

---



---

# max_eval_cost(3) MudOS v21c2 (5 Sep 1994)

## ******:

max_eval_cost() - ******* ***κķ*ʰ** (evaluation cost)

## ***:

void max_eval_cost()
* *** max_eval_cost()

## *÷*:

max_eval_cost() *********‘ (driver) *ж*ʰ*p*****
h*********Ãǰ, ******p*****e*’****.

## *o*:

catch(3), error(3), throw(3), error_handler(4),
set_eval_limit(3), reset_eval_cost(3)

## ****:

** Spock @ FF 97.Feb.18.** (printed 3/16/95)

# reclaim_objects(3) MudOS v21c2 (5 Sep 1994)

名称:

reclaim_objects - 清除残留在记忆体中的物件.

语法:

int reclaim_objects( void );

整数 reclaim_objects( 无参数 );

用法:

这个函式重复检查每一个载入的物件, 并尽力把残留在记忆体中的物件清除掉. 这样可以清出一些记忆体, 清理的数量多寡要看 mud 本身的程式是如何写的. 如果一个物件被其他物件里面的全域变数 (global variable)指标 (pointer) 指向到, 就会残留在记忆体中, 然後再被摧毁掉. 这个外部函式 (efun) 会传回变数遇到的被摧毁的物件数目.

参考:

destruct(3)

翻译:

Spock @ FF 97.May.24. (printed 3/16/95)

回到上一页

# replace_program(3) MudOS v21c2 (5 Sep 1994)

名称:

replace_program() - 把 this_object() (目前这个物件) 的程式置换掉.

语法:

void replace_program( string str );

无传回值 replace_program( 字串 str );

用法:

replace_program() 会把 this_object() 替换成这个物件继承的物件. 字串 str 是要进行置换的档案名称. 物件进行置换之後, 目前的物件就相当於继承物件的复制品. 只是保留目前物件的档案名称和全域变数 (global variable) 不变. 在目前的物件程式执行完毕以前, 不会进行置换的动作.

参考:

clone_object(3), new(3)

翻译:

Spock @ FF 97.May.27. (printed 3/16/95)

回到上一页

# reset_eval_cost(3) MudOS v21c2 (5 Sep 1994)

名称:

reset_eval_cost() - 重新设定剩下的执行耗费时间数字.

语法:

void reset_eval_cost();

用法:

reset_eval_cost() 把剩下的执行耗费时间数字设定成最高执行耗费时间数.

参考:

catch(3), error(3), throw(3), error_handler(4), eval_cost(3), set_eval_limit(3)

翻译:

Spock @ FF 97.Jun.2. (printed 3/16/95)

回到上一页

# set_eval_limit(3) MudOS v21c2 (5 Sep 1994)

名称:

set_eval_limit() - 设定执行耗费时间 (evaluation cost) 的上限值.

语法:

void set_eval_limit( int );

无传回值 set_eval_limit( 整数 );

用法:

以一个非零值的参数指定给 set_eval_limit() 函式, 则设定任何一个执行绪 (thread) 在发生错误之前所允许的最高执行耗费时间. 如果参数为 0, 则将目前的执行耗费时间计数器 (counter) 归零, 并传回目前执行耗费时间的上限值. set_eval_limit(-1) 传回剩下的执行耗费时间.

参考:

catch(3), error(3), throw(3), error_handler(4), eval_cost(3)

翻译:

Spock @ FF 97.Jul.21. (printed 3/16/95)

回到上一页

# set_reset(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� set_reset - ����h�������������´������趨 (reset) ��ʱ��.

## ���:

�� varargs void set_reset( object ob, int time );

�� ������������ ̈ ���̈ set_reset( ��� ob, ���� time );

## ���÷:

�� ̈趨 ob ������´������趨��ʱ��Ï���� time ̧̈́̈́̈́

# shutdown(3)�? MudOS v21c2�? (5 Sep 1994)

## ??????:

?? shutdown() - ?ر? mud ????????? (driver).

## ???:

?? void shutdown( int how );

?? ?  ???¸shutdown( ???? how );

## ???÷?:

?? ?????????????ƕر?????????.
??????Ï?????????ļ???ו?????????? kill
?????? ??????????????????????????????
?? shutdown ?????????????????????رjķ?'.
???? how¸?????????Ç???  ? exit()
?????????¸ ??????, ?? how Ĭ -1 ˠ,
??h??ر?????????????????????? script . ????
reboot (?????? mud) ????̣??Ö??'?? shutdown(), ??
halt ??ǎ( mud ??) Ö??'?? shutdown(-1) .??
????????Ç??????????ķ?'???? mud ?رjķ?',
??Ö????? script ?м?鷗??.
?? shutdown() ??Ȼ??h????Ç??õ?oγصĺ???. ??
simul_efun ?м??? shutdown()?? ??'?? exec(),
???????Ƨ??? shutdown() ???ɻ?ļ??.
??ȷ????????????? (master.c) ??XÇ?趨
valid_override(4) ?Ŀ? efun::shutdown() ????̣?'?ы?????.

## ???o?:

?? crash(4)  slow_shutdown(4)

## ??????:

?? Spock @ FF 97. Jul.23.?? (printed 3/16/95)

# time(3)���MudOS v21c2(5 Sep 1994)

## ������:

�time() - ������ʱ'��1970 ��h��h���ж����.

## ���:

int time( void );

time ( )

## ���-:

1970 ش ��h��h

(������ʱ)

,

.

## ���o:

ctime(3), localtime(3), time_expression(3), uptime(3)

## ������:

Spock @ FF 97.Jul.25. (printed 3/16/95)

# uptime(3)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� uptime() - ������h������������ (driver) ������������.

## ���:

�� int uptime( void );

�� ���� uptime( ( ) ( ) ( )

# ??????????õ????

---

??????????ęķ????į??????????????



---



---

# ??????????õ????

## ??????????ęķ????ļ???

# __INIT(4)�� MudOS v21c2�� (5 Sep 1994)

---


## ����:

__INIT - ·'����Ğ̄Ð̄�����.


## ·:

__INIT( void );

__INIT( ·  ··· );

## ·÷·:

··ǰ̄ǰ̄ǰ̄ǰ̄ǰ̄ǰ̄ (create)̋ ǰ̄···ô·'···. ··'·· (initialization)··
ȭǰ̄ǰ̄ǰ̄ǰ̄Ĺ···Ľǰ̄ǰ̄ǰ̄ǰ̄ǰ̄ǰ̄ǰ̄ǰ̄Í···q̄··,
··̈···Χ̨·  ·····Ɱ··'···Ĺ···. ··· __INIT ·····Ğ̄Ð̄·····.

## ····:

Spock @ FF 97.Aug.3.·· (printed 3/16/95)

---


[ص···hx]


---

# clean_up(4)�� MudOS v21c2�� (5 Sep 1994)

## ����:

clean_up - ��h������������ (interactive)
����ж��Ǐ��ô'���.

## ﷺ:

int clean_up( int inherited );

���� clean_up( ���� inherited );

## ﷺ:

������������ (driver) ï��h��ʰ����ûﷺ (inactive)
������������ clean_up() ����. ﷺ��ʰ��ij���,
�������ﷺ�ʰ������ (rumtime configuration file) ��.
clean_up() ���ɹ﷽h����¯ (flag) ,
ﷺ﷽﷽﷽

# ??create(4)?? MudOS v21c2?? (5 Sep 1994)

---

??????:

?? create - ????ij?'?? (initialization)

???:

?? void create( void );

?? ?  ???¨create( ?  ??? );

???÷?:

?? ÿh????????, ??Ö????h?? create() ????.
?? create() ???????, Ç???h????????ij?'??????
趨. ???e?????????????? create() ????. ??:??ʊ
ڑ??, create()?? ????Ï???? stock 3.1.2 LPmud
?????????. ?? 3.1.2?? ??,
???h??????h??'?ч??????????=?,
?Ŧ??????? create(). ?? MudOS ?oŦ?,
ÿh??????h???????? create() . ???,
??Ҳ??ǫ??hIIJ??¨??????. ????h????????
(parent object)?? ?e? create()?? ??????????Ķ???,
?ч?????????????????????Ķ???.
?????????h?????????????????? create().??
ч????, create()?? ??????????, ?¨
??????????. Ҳ???Ǹ?????e? create()
??????????e?parent::create(). (parent Ï??????)

???o?:

?? reset(4)  __INIT(4)

??????:

?? Wayfarer@Portals

# ��id(4)���� MudOS v21c2�� (5 Sep 1994)

---

## ������:

	�� id - ��`����� present() ����,
	��_h������������.

## ���:

	�� int id( string an_id );

	�� ���� id( ��_���� an_id );

## ���-�:

	�� present(3)�� ��  ���� (efun) ���� id()
	���ж�h��_�������������p�����_��� an_id��
	Ї¤������ (id) . ��� an_id��
	��Ǵ������ѕ¤������h, ��и��� 1, ������  ��0.

## ���o�:

	�� present(3)

## ������:

	�� Spock @ FF 97.Aug.5.�� (printed 3/16/95)

---



---

# init(4)���� MudOS v21c2�� (5 Sep 1994)

## ������:

�� init - move_object() ��������e init() ��'�'�� (initialize)�� ���� (verb) ��  ��(action).

## ���:

```
�� void init( void );

�� ��  ���_init( ��  ��( );
```

## ���-�:

�� ��mudlib ����A�� ������B�ʰ, ������(�move_object() ����)�� ������µĶ���:

�1.�� ��A�ǻ(living),�ǰ A�ǰ B�ǰ ������A ǰinit() ����.

2.ǰ ����A ǰp ï ,ǰ Bǰ ������(inventory) ����, ǰ л ǰ A ǰinit().

3.ǰ ǰA ǰ ǻ, ǰ A ǰ ����λǰ Bǰ ǰinit().

ǰ ǰ ʊǰ: hǰ ù ǰenable_command(3)ǰ ����ï .

ǰ h ǰ, hǰ init(4) ǰ add_command(3), ǰ ч ş Ķ ǰ(command).

## ���o:

•• reset(4), move_object(3), enable_commands(3), living(3), add_action(3)


••••••:

•• Spock @ FF 97.Aug.5.•• (printed 3/16/95)

---


•ص•••hx


---

# ��move_or_destruct(4) MudOS v21c2�� (5 Sep 1994)

## ������:

�� move_or_destruct - ��h������z����_����L������.

## ���:

```
�� int move_or_destruct( object dest );

�� ���� move_or_destruct( ��� dest );
```

## ���-�:

```
�� ���h������L��� (environment)��
������  ���, ,  ,,ô'���,
������������������. dest��
����Ç���  ����. ��� dest ������, ��  dest Ï
0. ��� dest e����û a  L  Y����  ����,
  X  h  .
```

## ���o:


�� destruct(3), move_object(3)


## ������:


�� Spock @ FF 97.Aug.5.�� (printed 3/16/95)

# reset(4)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� reset - ��h������������ά�� (self-maintenance).

## ���:

�� void reset( void );

�� ��  ���_reset( ��  ��� );

## ���:

�� ��ÿ������趨 (reset)��ˇ ��  (ι��ι������趨
����ą̈́���Ĵ mud��  ���趨,
ѕ����L��ÿ����Cʰh��) ,
������Þ������������������ reset()��
����. ������������������ʰ, ��  options.h��
ж����  LAZY_RESET,��
˛����eЧ������Ê�Ż����  reset(). ���h��,
û�õ������Ť���χ����  (swap file)��
˛����  ������趨. reset() ˛����ن鱦
˛˛˛˛˛˛p���ij˛˛˛˛˛,
˛˛˛ж˛Ç˛˛Ç˛˛˛²˛˛h˛˛.

## ˛˛˛o˛:

˛˛ set_reset(3), create(4)

## ˛˛˛˛˛˛:

˛˛ Spock @ FF 97.Aug.5.˛˛ (printed 3/16/95)

# ??????????õ????

## ??????????eķ????ļ???




ص???hx

# catch_tell(4)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� catch_tell - ̣ş̣h������′���  ����Ľ��� (interface).

## ���:

�� void catch_tell( string message );

�� ̣ ̣ ̣

# logon(4)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� logon - ��'�� (initialize) h����  ��  (login connection).

## ���:

```
�� object logon( void );

�� ��  logon( ?  ??  );
```

## ���=?:

```
�? ?????? (master) ?e? connect()
??????ص????????? logon().
?'??????µ?'?????趨??.
```

## ???o?:

```
?? connect(4)
```

## ??????:

```
?? Spock @ FF 97.Aug.5.?? (printed 3/16/95)
```

# net_dead(4)�� MudOS v21c2�� (5 Sep 1994)

## ������:



## ���:



## ���÷�:


�� ���h���ɻ�������(��������) ɔŒ���(�����ˈ���·������(net dead)��ˇ), ������������ʔ������ net_dead(), ô������ժ��л�̈ ̈ ̈ ̈ ̈ (clean up)��������˝������������ŋ̈. ��C��, ������̈ ̈ ̈ ̈ 

# process_input(4) mudOS v21c2�� (5 Sep 1994)

## ������:

�� process_input - ⁄ MudOS⁄ ⁄ ⁄ 

# receive_message(4) MudOS v21c2�� (5 Sep 1994)

## ������:

�� receive_message - ̣ message() ̣ ̣ ̣

# receive_snoop(4) MudOS v21c2�� (5 Sep 1994)

## ������:

�� receive_snoop - ʊɣ'�ɔ����L��( (snoop) ����.

## ���:

```
�� void receive_snoop( string message) ;

��  ���¸receive_snoop( ¸���( message
);
```

## ���÷�:

```
�� ������������������ʰ, ��  options.h��
ж��( RECEIVE_SNOOP,
��h��'���  ������h��'����ʰ,
���п��ч����ɹл  ��������e
receive_snoop()��  ����. ڳ'�����,
������Ղ-�����Щ����. receive_snoop()
��������,'������Щ�����  ��ˢ receive()
����.
```

## ���o:

```
�� catch_tell(4), receive(3), receive_message(4)
```

## ������:

```
�� Spock @ FF 97.Aug.5.��  (printed 3/16/95)
```

# telnet_suboption(4) MudOS v21c2�� (5 Sep 1994)

## ������:

�� telnet_suboption - ���� telnet ��  suboption.

## ���:

�� void telnet_suboption( string buffer );

�� ��  ���¸ telnet_suboption( ¸���  buffer
);

## ���÷�:

�� ��  mudlib ����Ĵ��������, ¸ţ�����
(interactive) ������ô˙���, ����  SE telnet suboptions
��Ï������. ����:
δ  ˙���¸  ßų¸�����˙.

�� buffer��  ĵ�h����  (byte)
c����h������������¸ (type descriptor),��  ��
TELOPT_TTYPE.��  ĴĴ����¸h��  procession option,
��  TELQUAL_IS.��˜ ������������ص�����.
�����ßų¸  ����¸  ���  (terminal type call)ʰ,
��Ш¶���  ���  terminal_type() ��������.

## suboptions:

### #define TELQUAL_IS������
0������������������  // option ��...
### #define TELQUAL_SEND��  1
// ˫  options
### #define TELQUAL_INFO��  2
// ENVIRON: informational version of IS
### #define TELQUAL_REPLY 3
// AUTHENTICATION: client version of IS

`#define TELQUAL_NAME`
`4` // AUTHENTICATION: client version of IS


:

`Spock @ FF 97.Aug.5.` (printed 3/16/95)

---

hx

---

# terminal_type(4) MudOS v21c2�� (5 Sep 1994)

## ������:

�� terminal_type - ��¸mudlib '��  ��ŭ¸����  (terminal type).

## ���:

�� void terminal_type( string term );

�� ¤  ���¸terminal_type( ¸���  term );

## ���÷�:


�� ï'������趨ŭ¸����ʰ,
¤ų������������ô'���. ������  telnet ����  (negotiation)��  ʰ������.
���'���  ¬jj���û¤л¤Ö (����¸¸����  telnet),
����Ƨ���  ��ô'���.


## ������:


�� Spock @ FF 97.Aug.5.��  (printed 3/16/95)



[ص]hx

# write_prompt(4)�� MudOS v21c2�� (5 Sep 1994)

---

## ������:

�� write_prompt - ������������ (parser)
����’��’���� (prompt)�ħ, ��̨��ô’���.

## ���:

�� void write_prompt( void );

�� ��  ���̠write_prompt( ��  ��  );

## ���÷:

�� ������������������ write_prompt(),��
��Ų������’���Ÿ���’ħ, ������  ��ô’���.
������������ input_to (������) ��  ed
(��  ����)��ˬħ, ������  ���  write_prompt()

## ������:

�� Truilkan@TMI

## ������:

�� Spock @ FF 97.Aug.5.�� (printed 3/16/95)

---



---

# ??????????õ????

## ??????????ęķ????ļ???

valid_seteuid.4
valid_shadow.4
valid_socket.4
valid_write.4
valid_errors.4

---

hx

---

# author_file(4)�� MudOS v21c2�� (5 Sep 1994)

## ������:



## ���:



## ���÷:



## ���o:



## ������:

# compile_object(4) MudOS v21c2�� (5 Sep 1994)

## ������:

�� compile_object - ��ş������ (virtual object) ��
mudlib�� ����

## ���:

�� object compile_object( string pathname );

�� ��� compile_object( ��ˍ��� pathname );

## ���ˍ�:

�� ⁄ mudlib
ˏ'h
compile_object().�� ������ˎ,
mudlib�� ���� call_other("/obj/file.r", "some_function")
�� new("/obj/file.r") ʰ, ˊ
(master.c) ��e compile_object("/obj/file.r"). ⁄ /obj/file.r⁄
Ï. ⁄ mudlib⁄
⁄ (associate)⁄ ⁄Ï /obj/file.r
⁄ ⁄ compile_object()⁄ ⁄ 0. ⁄ mudlib
⁄, ⁄ /obj/file.r⁄ ⁄. ⁄
compile_object() ⁄h⁄ˍ⁄ˇ⁄⁄,
⁄, ⁄ file.r.c⁄
⁄ˍ⁄Þ⁄. ⁄ ⁄ʰ,
⁄ compile_object() ⁄ص⁄.

## ������:

�� Spock @ FF 97.Aug.6.�� (printed 3/16/95)

# connect(4)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� connect - Ï�µ�'����ą��h�����.

## ���:

�� object connect( void );

�� ��� connect( ( ��� );

## ���÷�:



## ���o�:

## ������:



---

# crash(4)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� crash - ������������ ��(crash)ʱ,
�¸ ������������ (master.c)�� ��Ĵ'���.

## ���:

�� void crash( string crash_message, object command_giver, object current_object );

�� ¸ ¸ ¸

# creator_file(4)�� MudOS v21c2�� (5 Sep 1994)

---

## ������:

�� creator_file - ��h���
´������������'����ʁ������ (user id, uid).

## ���:

�� string creator_file(string filename);

�� ��ˌ��ˌ creator_file( ��ˌ��ˌ file_name );

## ���÷�:

�� ÿ�δ���h���µ����ʰ,
������������������ (master object) ��creator_file()
����. ������ˌ¦������ creator_file()
����Ψh�IJ���. ��creator_file
���ص��ˌ����Ǵ�����趨��'����ʁ������.
������������������ʰ, ��options.h
������AUTO_SETUID ,
����ʰh��ʁ������ï�������Ч'����ʁ
(effective user id, euid)

�� Ψh������: ������������������ʰ,
������ AUTO_TRUST_BACKBONE , ��creator_file()
���ɛÑ'����ʁ������ (backbone
uid)������������ get_bb_uid() ���ص����,
������'����ʁ����ɀ���Ч'����ʁ����

## ���o:

�� seteuid(3), new(3), clone_object(3), call_other(3)

## ������:

�� Spock @ FF 97.Aug.6.�� (printed 3/16/95)

# domain_file(4)�� MudOS v21c2�� (5 Sep 1994)

## ������:


�� domain_file - ж h ̣ ̣ ̣

# epilog(4)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� epilog -
����h�������������ļ����Z������ (array),
��������e��ļ�Ï����Þĭ����Ą̨̨̨

# error_handler(4) MudOS v21c2�� (5 Sep 1994)

---

## ������:

�� error_handler - ������������������í���.

## ���:

�� void error_handler( mapping error, int caught );

�� ��  ���̲error_handler( ÿ�� error, ����
caught );

## ���÷�:

�� ·����� mudlib
������������������������. ÿ����� error
������Ï:

```
�� ([
���� "error"���� :
ò����,������������������ // ��������
���� "program" : ò̲���,������������������ //
����ij���
���� "object"�� : ���,������������������ //
Lǰ�����
���� "line"������ :
����,������������������ //
������������ٱ���
���� "trace"���� : ÿ��*������������������ //
ب  ���
�� ])
```

��  ��ب  ÿh�ж���ÿ��, ������������:

```
�� ([
���� "function"�� : ò̲���,������������������
```

// ������
���"program"��: 
-���,
���"object"��: 
���,
���"file"��: 
-���,
//
������Ǐ��¦
���"line"��: 
// ����
�] )

�� ������ catch() ����γ��, caught ��¯ (flag) Ï 1.

## ���o:

- catch(3), error(3), throw(3), log_error(4)

## ������:

- Beek

## ������:

- Spock @ FF 97.Aug.6.�� (printed 3/16/95)

---

# flag(4)���� MudOS v21c2�� (5 Sep 1994)

---

## ������:

	�� flag - ������������ħ, ���� mudlib
	��ˍ�����¯���ض (flag).

## ���:

	�� void flag( string );

	��  ���ˍ flag( ˍ��� );

## ���÷�:

	�� ������������ħ,
	�������ъєт�������������ˍ -f w��,
	ÿh������������������w  ̨p�� flag()
	IJ.

## ������:

	�� Spock @ FF 97.Aug.6.�� (printed 3/16/95)

---

ض���hx

---

# get_bb_uid(4)�� MudOS v21c2�� (5 Sep 1994)

---



## ]

]

## ]

]

## ]

]

## ]

]

---


[ ]

# get_root_uid(4)�� MudOS v21c2�� (5 Sep 1994)

---

## ������:



## ���:

	�� string get_root_uid( void );

	�� ۪���۪ get_root_uid( ۪  ۪۪۪ );

## ���÷�:

	�� ۪۪۪۪۪۪۪ÿ۪۪۪۪۪۪۪۪۪۪۪۪۪۪۪ (master object)�� ʰ,
	۪۪۪۪ô¯۪۪۪۪L۪۪۪۪۪۪۪۪۪۪۪۪p۪۪Ģ۪۪۪۪۪۪ģ
	۪۪ģ۪ø۪ mud۪۪ ۪۪۪۪۪۪ж۪۪۪۪ root
	'۪۪۪۪ʁ۪۪۪۪۪۪. ۪¯۪۪۪۪۪۪۪  ۪۪h۪۪۪ۥ۪۪۪,
	Σ۪۪ ROOT ۪۪۪۪۪۪ۥ۪۪۪.

## ۪۪۪o۪:

	۪۪ get_bb_uid(4)

## ������:

	۪۪ Spock @ FF 97.Aug.6.۪۪ (printed 3/16/95)

---



---

# get_save_file_name(4) MudOS v21c2�� (5 Sep 1994)

## ������:

�� get_save_file_name - ' ����������ʰ, ������ Ĺ������.

## ���:

�� string get_save_file_name( string );

�� ˍ���˙ get_save_file_name( ˍ���˙ );

## ���÷�:

�� ��h������ڳ ˙˙˙˙б ˙ļ˙ʰ˙˙˙ Ļ˙, ed()�� ˙˙˙˙˙˙˙˙˙ get_save_file_name().�� ˙ˋ˙˙˙Ö˙˙˙˙˙˙ ˙˙˙ļ˙˙˙˙˙˙˙˙˙, ˙˙˙д ˙˙˙˙˙ļ˙.
�� ш��:
�� ˙ˋ˙˙˙˙˙ǰ˙˙Ї get_ed_buffer_save_file_name().

## ���o˙:

�� ed(3)

## ������:

Spock @ FF 97.Aug.6.�� (printed 3/16/95)

# log_error(4)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� log_error - ����  ¼����.

## ���:

�� void log_error( string file, string message );

�  ��log_error( ���file,
message );

## ���-:

(compilation)  ħ  κδ,
Ï file
message ,
(master object)  e log_error()
. ô  ħ, log_error()
ï  ¼.  log_error()  ,
ü¼Çд  к  Ш ¦ .

## ���o:

error_handler(4)

## ������:

Wayfarer@Portals

## ������:

Spock @ FF 97.Aug.6. (printed 3/16/95)

# make_path_absolute(4) MudOS v21c2�� (5 Sep 1994)

## ������:

�� make_path_absolute - ת������·������ (relative path name).

## ���:

�� string make_path_absolute( string rel_path );

�� ˙���˙ make_path_absolute( ˙���˙ rel_path );

## ���÷�:

�� ed()��
ת������ô’������ת���¦������·������Ï
(absolute path name) .

## ���o:

�� ed(3)

## ������:

�� Spock @ FF 97.Aug.6.�� (printed 3/16/95)

# object_name(4)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� object_name - ����������ô¨�����¸��h�������������.

## ���:

�� string object_name( object );

�� ¸��¸ object_name( ��¸ );

## ���÷�:

�� ¸¸ sprintf(3) ����Ç3��h������ġ¨¸ʰ, ����ô¨¸¸¸. ¸¨¸¸¸Ö����h¸¸¸¸¸¸, ¸¸¸¸¸¸¸¸Ö¸¸¸¸¸ object ¸¸¸¸¸¸ (¸¸h¸¸'¸¸¸ ¸¸¸¸¸¸).

## ¸¸¸o¸:

�� file_name(3)

## ¸¸¸¸¸¸:

�� Spock @ FF 97.Aug.6.¸¸ (printed 3/16/95)

# preload(4)�� MudOS v21c2�� (5 Sep 1994)

---

## ������:

�� preload - ��h�����Ţ�������ڲ.

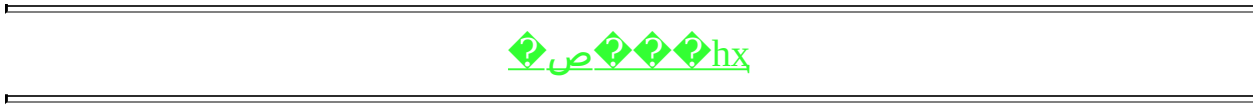
## ���:

```
�� string *epilog( int load_empty );
�� void preload( string filename );

�� ̲���* *epilog( ̲���* load_empty );
��  ̲ ̲ ̲ preload( ̲���* filename );
```

## ���÷:

```
�� ̲̲̲
```



# privs_file(4)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� privs_file - ï�´������_��h��_����_��� (privs string).

## ���:

�� string privs_file( string filename );

�� ̲��� privs_file( ̲��� filename );

## ���÷:

�� ̲̲̲̲į̲̲̲̲̲̲̲̲ʰ, ̲̲̲̲̲̲̲̲̲̲̲̲̲̲̲̲ (master object) ̲e̲ privs_file() ̲̲̲̲.
̲̲̲̲̲̲į̲̲̲̲̲ï̲̲̲̲̲, ̲̲
̲̲ privs_file()̲̲
̲̲̲ص̲̲̲̲̲̲̲̲̲̲̲̲̲̲̲̲

# retrieve_ed_setup(4) MudOS v21c2�� (5 Sep 1994)

---

## ������:



## ���:



## ���÷:



## ���o:



## ������:



---



---

# save_ed_setup(4) MudOS v21c2�� (5 Sep 1994)

## ������:

�� save_ed_setup - ����h��'���  ı  ������趨 ����ę趨.

## ���:

```
�� int save_ed_setup( object user, int config );

�� ���� save_ed_setup( ��� user, ���� config );
```

## ���÷�:

```
�� ed()��
   ���ð'����Ղ���h��'���  ı  ������趨
   ������趨 (����h�������). ����ı�ʰ,
   ę'���Ö����  1��  ���根 (�  : ������).
   ��ˇ  ¸ 0��  ��α��. (¸  : ����).
```

## ���o:

- retrieve_ed_setup(4)

## ������:

Spock @ FF 97.Aug.5.�� (printed 3/16/95)

---

# slow_shutdown(4) MudOS v21c2�� (5 Sep 1994)

## ������:

�� slow_shutdown - ��˛mud�� Ľǰ������ر����ڀ (slow shutdown) ŕĹ�����.

## ���:

�� void slow_shutdown( int minutes );

�� ˙ ˙ ˙

# valid_asm(4)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� valid_asm - ���� LPC->C ����������p���'��
asm { }

## ���:

�� int valid_asm( string file );

�� ���� valid_asm( ?_��? file );

## ���÷�:

�� ?????????????ћ???????? LPC_TO_C w??,
?? LPC?? ?Ђ???????h??????? asm { }??
?ű?ɔ????? valid_asm()?? ????.
????'???????? 0,
???жϱ?????_??????????.

?? asm { } ?ű???÷?????:

?? asm {
???? < C ?Ђ???? >
?? }

?? ???h??, ??????? asm { ʃ ?????? C??
?Ђ????.

## ???o?:

?? valid_compile_to_c(4)

## ������:

?? Spock @ FF 97.Aug.6.?? (printed 3/16/95)

# valid_bind(4)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� valid_bind - ���� bind() ��������;.

## ���:

�� int valid_bind( object binder, object old_owner, object new_owner );

�� ���� valid_bind( ��� binder, ��� old_owner, ��� new_owner );

## ���-�:

�� ��h������'�� bind() ������ij������Í������ô'������. ��� valid_bind() ���� 1 ������ bind() ĶĶ, ���� 0�� ��������.

## ������:

�� Spock @ FF 97.Aug.6.� (printed 3/16/95)

# valid_compile_to_c(4) MudOS v21c2�� (5 Sep 1994)

## ������:

�� valid_compile_to_c - ����h�������������ħ�� (runtime) ��, ȹ������ LPC->C ķ̀ʻ���ʻ (compile).

## ���:

�� int valid_compile_to_c( string file );

�� ʻʻʻʻ valid_compile_to_c( ʻ̣ʻʻʻ file );

## ���÷�:

�� ʻʻʻʻʻʻʻʻʻʻʻʻʻʻʻʻʻʻʻʻʻħ, ʻʻʻʻʻʻʻʻʻʻ LPC_TO_C ʻʻ RUNTIME_LOADING,ʻʻ ʻʻʻʻʻʻʻʻЛЪ.Cʻʻʻʻβij ʻʻʻ (ʻʻʻāā.cʻʻ) ħ, ʻʻʻʻʻʻʻʻʻʻʻʻʻʻʻʻ (master object)ʻʻ ʻʻ valid_compile_to_c() ʻʻʻʻ. ʻʻʻʻ̀ʻʻʻʻʻʻʻʻ 1, ʻ̊ʻʻʻɨʻʻʻÏ Cʻʻ ʻʻʻʻ. ʻʻʻʻʻʻʻʻ 0 , ʻ̊ʻʻʻʻʻÏʻʻʻʻ LPCʻʻ ʻʻʻ. ʻʻʻʻʻʻeʻʻʻ̣ʻʻʻʻʻʻʻʻʻʻʻβʻġʻ.Cʻʻ.

## ȹ������:

�� Spock @ FF 97.Aug.7.ʻʻ (printed 3/16/95)

# valid_hide(4)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� valid_hide - ������h������������������ (hide) ؋J������������������.

## ���:

�� int valid_hide( object ob );

�� ���� valid_hide( ��� ob );

## ���-�:

�� ������������ (master.c) ���� valid_hide() ؋J؋h؋؋؋

# valid_link(4)�� MudOS v21c2�� (5 Sep 1994)

---

## ������:



## ���:



## ���÷�:



## ���o⁄:



## ������:


�� Spock @ FF 97.Aug.7.⁄⁄ (printed 3/16/95)


---



---

# valid_object(4)�� MudOS v21c2�� (5 Sep 1994)

## ������:



## ���:



## ���÷�:


�� ����h������ˇ��, ̈́̈́̈́

# valid_override(4) MudOS v21c2�� (5 Sep 1994)

## ������:

�� valid_override - ����′�� efun:: ������.

## ���:

```
�� int valid_override( string file, string efun_name );

�� ���� valid_override( ̣���  file, ̣����  efun_name );
```

## ���÷:

```
�� ������������ (master.c) ̀м��  valid_override()
���Ч���  efun::��  ����ǰ���  (prefix)
��′������.  ̀̀̀̀̀̀̀̀̀̀̀̀Ш���ǰ̀̀̀̀̀̀
efun::  ̀Í���ћ,  ̀̀̀̀̀̀̀̀̀̀̀̀̀̀̀̀  valid_override().
���  valid_override()  ����  0,  ̀̀̀̀̀̀ц��.  ̀̀̀h̀̀,
valid_override()̀̀
̀̀̀̀̀̀hЩ̀̀Ç̀̀̀  ̀̀̀̀̀̀̀̀̀̀̀  efun::
̀̀̀̀ǰ̀̀̀  ̀̀̀  ̀̀̀Щ̀  ̀̀̀̀̀̀̀̀̀̀̀
(Ç̀  ̀  ’̀  ̀̀̀̀ћ,  ̀̀̀̀̀̀̀̀̀̀ğ̀̀̀  ̀̀̀̀
(simul_efun)̀̀  ̀̀̀  ̀.  ̀̀′̀̀  valid_override()
̀̀̀̀Щġ̀̀̀  ̀̀̀̀̀̀̀̀̀̀̀̀̀̀̀̀ЂѴĘ̀  ̀
(overriden) ).

̀̀  ̀̀̀̀̀ш̀̀′̀̀  ̀̀  3.1.2̀̀  ̀̀  efun::
̀̀̀̀̀̀¸̀̀Ç̀̀̀̀̀̀̀̀̀̀  master.c
̀м̀̀̀̀̀̀̀̀̀̀̀̀:
̀̀
```

```
    ̀̀ int valid_override(string file, string efun) { return 1; }
̀̀  ̀̀̀̀̀̀h̀̀̀̀̀̀ѕэ̀̀  valid_override()  ̀̀̀̀:
̀̀
    ̀̀ int valid_override( string file, string name )
    ̀̀ {
```

```
        ???? if (file == "/adm/obj/simul_efun") {
            ?????? return 1;
        ???? }

        ???? if (name == "destruct")

            ?????? return 0;
        ???? if (name == "shutdown")
            ?????? return 0;
        ???? if (name == "snoop")
            ?????? return 0;
        ???? if (name == "exec")
            ?????? return 0;
        ???? return 1;

    ?? }
```

??????:

?? Truilkan@Basis

???o?:

?? valid_object(4), function_exists(3)

??????:

?? Spock @ FF 97.Aug.7.?? (printed 3/16/95)

---



---

# valid_read(4)�� MudOS v21c2�� (5 Sep 1994)

## ������:



## ���:

	�� int valid_read( string file, mixed user, string func );

	�� ����  valid_read( ę��  file, ���  user, ę���  func );

## ���÷:

	�� ÿ€h  ′  ��Ɔďę̌ļ,
	valid_read()
	Ł����ĶďĘ. ę̀����IJ���  file
	Ïļ̌, user  Ï���Ɔďę̌ˬ, func
	Ï̌ɵ���  valid_read()  Íę̌. ���  valid_read()
	���  ,̣ط̌ď.

## ���o:

	valid_write(4)

## ������:

	Spock @ FF 97.Aug.7.  (printed 3/16/95)

# valid_save_binary(4) MudOS v21c2�� (5 Sep 1994)

---

## ������:

�� valid_save_binary - ����h������p���Ղ-�������Ղ-����.

## ����:

�� int valid_save_binary( string file );

�� ���� valid_save_binary( ��-��� file );

## ���-�:

�� ����������ʰ, ��������
ALWAYS_SAVE_BINARIES; ����h������ж�����
#pragma save_binary,�� ɔ����ļ�����
��Ї�������
valid_save_binary().��
����˙������ 1, ������. ��ʊ̃Ť�����.

## ������:

�� Spock @ FF 97.Aug.7.� (printed 3/16/95)

---

ص���hx

---

# valid_seteuid(4) MudOS v21c2�� (5 Sep 1994)

---

## ������:



## ���:



## ���÷�:



## ���o̲:



## ������:



---



---

# valid_shadow(4)�� MudOS v21c2�? (5 Sep 1994)

## ������:

valid_shadow - ������Щ�������ЇИЎ (shadow).

## ���:

int valid_shadow( object ob );

valid_shadow( ��? ob );

## ���-�:

h Ї ob ИЎ (' shadow() (efun) ) , e valid_shadow() ����. ob Ї'IJ, ob previous_object() ИЎ ob, valid_shadow() Ö 0 , shadow() 0. valid_shadow() 1, ИЎ.

## ���o:

shadow(3), query_shadowing(3)

## ������:

Spock @ FF 97.Aug.7. (printed 3/16/95)

# valid_socket(4)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� valid_socket - ���� socket ˙ ˙ ���� (efunctions).

## ���:

�� int valid_socket( object caller, string function, mixed *info );

�� ���˙ valid_socket( ��˙ caller, ˙˙˙ function, ˙˙˙ *info );

## ���-˙:

˙˙ ÿh˙˙ socket ˙ ˙ ˙˙˙˙˙˙ĭj, ˙˙˙˙˙ęę˙˙ valid_socket(). ˙˙˙ valid_socket() ˙˙˙˙ 0, ˙˙˙ socket ˙ ˙ ˙˙˙˙ţ˙˙. ˙˙˝, ˙˙˙˙ 1˙˙ ˙˙ı˙. ˙˙h˙˙˙˙˙˙ caller ˙ǵ˙˙ô˙ socket ˙ ˙ ˙˙˙˙˙˙˙˙˙˙. ˙Ĵ˙˙˙˙˙˙˙ function ˙Ǵ˙ socket ˙ ˙ ˙˙˙˙˙˙˙˙˙˙ (˙˙˙˙ socket_write()˙˙ ˙˙ socket_bind() ).
˙˙˙˙˙˙˙˙˙˙˙˙˙h˙˙˙˙ЩЈ˙˙˙˙˙˙.
˙˙˙˙˙˙˙˙Щ˙Ļ˙, ˙˙˙˙˙˙˙˙ĵ˙h˙˙ ˙˙ (element) ˙Dzo˙˙˙˙ļ̃˙˙˙˙ (file descriptor being referenced).˙˙ ˙Ĵ˙˙˙ ˙˙˙˙ÿ˙д˙ socket ˙˙˙˙˙˙ (˙˙˙˙˙˙˙˙).˙˙ ˙˙˙˙˙˙˙ ˙˙˙Ǵ˙ socket ˙˙Ƣ˙˙λ (address of the remote end)˙˙ (˙˙˙˙˙˙˙˙˙).˙˙ ˙˙˙к˙ ˙˙˙˙˙˙˙ socket ˙˙ص˙ (associated) ˙˙˙ў˙˙˙˙˙ (port number).

## ˙˙˙˙˙˙˙:

˙˙ Spock @ FF 97.Aug.7.˙˙ (printed 3/16/95)

# valid_write(4)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� valid_write - ���h������p���Ę��д��h���ļ.

## ���:

�� int valid_write( string file, mixed user, string func );

�� ����� valid_write( ę���file, ��user, ę����func );

## ���÷:


```
�� ÿ�ε�h  ′  ���д���ļ,
������������������
�� ����valid_write()�� ĻL����ˇ�д��Ę��.
ę`����IJ
�� ��file Ï�ļ�����, user��
Ï����д��ˇ������, func
�� Ï��ө���valid_write()�� ę́.
�� valid_write() ���  ę ,ę̀ę̀ط̀д̀.
```


## ���o:


�� valid_read(4)


## ������:


�� Spock @ FF 97.Aug.7.�� (printed 3/16/95)

# view_errors(4)�� MudOS v21c2�� (5 Sep 1994)

## ������:

�� view_errors - ���h��'�����p ��������Ш.

## ���:

�� int view_errors( object user );

�� ���� view_error( ��� user );

## ���÷�:

�� ������������ĥ, options.h ءر��MUDLIB_ERROR_HANDLER�� w��, h��'������ۈ, ɔ���� view_errors( user ). ����'������ˌ ط ���'��ɔ��ɻ�h����þ�ĵ�����Ш. ��ʺ, ���ʒ��Ղ��ĵ�����Ш.�� (�: ¢���ă�������, ��ˬ��ʦ. ��h�����Ш)

## ���o�:

�� MUDLIB_ERROR_HANDLER(3)

## ������:

�� Spock @ FF 97.Aug.7.�� (printed 3/16/95)

---

# LPC ?к???

---

## ??????????eķ????ļ???



---



---

# ō������������ļ (global include file)

---

������ļ (config file) , ���h��w��Ï Global include file (����Ï: õ�������ļ). ����д-���, ������ļ����Ƨ�������ļ��Ŀ�и��, ������e��ļ����˝��� #include ������ļ�����.

������hШĮ�������������e����, ��Ïh������õķ���.

---

����: Spock @ FF 97.Aug.7.

---

ص���hx

---

# LPC ???? (lpc)

* LPC ??ô ?

LPC ??д MudOS (?? LPmud) ??????'?õ??????. LPC ?ă?Lars Pensj¦ C????????д. ??????????, ??Ҳ????µ? LPC ???? C ??? (syntax) Ï ???????. LPC ?ş?? C ???Ъ? while ижи (loop)?? for ижжи if ??? (statement)??switch ??輛??д????? sscanf ??????? (integer) ?????????? (LPC Ҳ?ş??????? C ????????  ??????????, ???????? (object) ??ў?? (mapping) ) . LPC '?? C ???  ???庸 ??????ú?????????????????. ??ш??, LPC ??????????????????? C ????????ş????????, ???ż???IJ???. ??þ??????????o? LPC ???  ??????Ћʊű?Ľ?ʌ?ļ?.

??????? LPC ?? Ć ???hЩ?????ˇ??:

LPC ??????, ?????Ç??Ï main ?Í???
(??Ć??h??????? create ?Í???).

??Þ?????????? (gamedriver) ?ş????  ????? (efun)?? (???Ïєт????? (system calls) ), ?? C ??????? (library) (?? libc.a) ??????ş?????ő??.

LPC ??û?? malloc(). ???????h?? allocate( ????? value ) ?  ??????????Ï????? (arrays) ??????ш?? .?ڜ??, allocate ????? (argument) ?ĵ?λ???????? (bytes), ?????  ?? (element) ?????Ł.

?ڜ治
?????ı????иŵ? (never explicitly deallocated).
??Þ??????????????h??????  ??o?
(reference) ???ˁ?.
???o???

(reference count) ��Ï�� (û����к������д ����Ł���) ,
ə������������ ˌ���′õ��ڑ̃n̈
(��������).

LPC ̈ë̤̈̈̈̈̈̈̈̈̈̈̈̈̈̈̈ŀŁƆχ̈ BASIC. ̈̈̈̈̈̈̈̈ C
̈̈̈̈̈̈̈̈̈̈̈̈̈̈̈̈

# ��щ������ (message document)

---

message() ��щ������

---

message() ��MudOS ��, ������ż��  ����(efun).
�'��������;��'��щ���  ������,
���ş��щ�������ı.

������ message ��'��ˍ��:

---

��:

void message( mixed class, string message, mixed targets, mixed exclude);

�  ���
message( ��class, ��message, ��targets, ��exclude );

message() ������κν������ѿ������ receive_message( ��class, ��message ) ����(target ��������æ������щ������).�exclude ���r�������щ������. ������,
������щ��Ł������.

message ����Ç������щ��.

class ����щ������
(����щ)
hШ�������: combat (u��)��shout (ź)��emergency (Σ��) ηˌ.

target ·æ������щ.
������h������ļ����ż��,
Χ������h������ (pointer),

������Ʉ�������ą����2ę�������� (array).

exclude ĐDz����ɹ���Ɯ�������.
������h����ę����Ʉ����ę�������.

---

message() ��������Ç�ĵ⊾Ö���� class ����.
��������ɹ′����,
�������������h���Ķ ����Ø������Ɯ,
��������Ɯ�������Ɂ-���. class
�������message ę������Ɯ�ķ���Ï��.
h��’ą ķ���Ö����• shout (ź)��say (/)��write
(��’)��tell_object (Ɯ) (tell_object
ġ���• ����• (simul_efun) e tell_object() ,
㳀ᴛ���• ����).

,
ę������Ç������h���• Ķ
(ź)Ĺ). ′ (ĐĐ)
, ĐÇĐ receive_message() ....
ę������h����• ę•••••• (•••• mudlib ,
Ö����hШą•••••Ⱶ•••ı•••••)

```
void receive_message (string msg, mixed class)
{
    receive(msg);
}
```

ę•••••ɹ message()Đ ę•••••••••e•••Ɯ,
Đ• ̀••••’Đ′••••. ••••, ę•••••••
h•••Ķ •••:

```
/* ȳ’Đ muffle ••••••, ĐĐĮĐ muffle û㠪•••ɦ, Đ
member_array() ę•••••••••Ɯ. */

    string *muffle = ({});

// •••• muffle_class() ••••••ɦ•••ɔĐę••••••.
```

```
        int muffle_class ( string arg )
        {
                muffle += ({ arg });
        }

/* receive_message() ?? message() ???õ???Щ?????.
?ڲ?? muffle ????????p??и? class ?????????,
û?oи???. */
??

        void receive_message ( string msg, string class )
        {
                if ( member_array( class, muffle ) == -1 )
                        receive(msg);
        }

?????ج??Ч???,
????ض?????????Щ???????
(?????ã??ź???),
???ź?????Щ??Ɔ????????`????,
??????????Щ??????????`.

????,
?ڲ?т??
mud ???????, ?????????'?? shout()
????Щ?????ź?. Σ??~, mud
єт????Ш???????e?'?????????? mud
?????????Ғ?ر?.
???h??,
        ??? echo ??. ?? echo ????'?? shout().
??~???'???ź?????'??????????????Ç????Щ
????????α?'??Щ????Ç?  ???.
????,
????Ǫ??????????Щ????ï?µ?
broadcast (?虧) ????.
?????????????ÿh???„??????????????Ç??Щ
????Ö????'????'?ù
虧
????.
```

```
varargs int emote (object emoter, string self_message, string
        other_message, mixed emotee, string target_message, string modifier);
```

emoter - ...
self_message - ...
other_message - ...
emotee - ... (... kick huthar)
target_message - ...
string modifier - ... (... smile happily, cheerfully ...) - ...
(...): ...

������, Ç�V��  ���′�  ���Щ�������  ���.
���L�������h�� combat ���  h�� stat (���)
���  h�� room_description (�������) ���  h�� help
(�������) ����. ҉ĉ’ǰ, ������дh���µ
receive_message().

```
int has_smart_client;

void receive_message (string msg, string class)
{

    if (member_array(class,muffle) == -1) {
        if (has_smart_client)
            receive (class + " : " + msg);
        else
            receive (msg);
    }
}
```

��. ������������Щ����ıô.
���h��′������������ has_smart_client > 0,
�����e���Щǰ涼����Þ�������. ����,
�������дh�������Щ
�����v  ������h���Ч,
′������h���Ч, u����Щ�����h���Ч,
҉Ղ-����. ������  ҉h��″��,
������’Ŀǰ��ķ (��Ï҉����  ��ĥ,
ҷ҉h�� room_name (������) ������Щ) .
������ heart_beat() ����,
��″������’Ŀǰ.
��Щ������Щ������_‘��’���ů␍҉′҉‘

������′҉ô ̃дh  ҉c҉v����û
BSX c mud ���û҉ ̊ MudOS ��
message()  ̊گ.
ĶX҉Ղ҉h҉C҉ĵc҉ (bitmap) ���  BSX
Ķ c (polygon-based line drawing). message()
;.

������������/������ӧ��,
�����и�������������.
��Ćÿ���˷Ç�Ł�д�������Щ��Щ,
Хû����д������õ��û��˚��(client)
Ł������Щ��Щ����ij���.
h����Щ������д������, ���(mudlib
��ʂô˚������˚������?ð,
˚˚˚˚˚˚˚˷˚˚˚ı˚˚˚ص˚˚˚˚.
˚˚Щ ˚˚Ћ˚˚˚h˚˚˚ ˚˚˚ЩЭ˚˚˚ô˚e˚˚˚,
˚˚˚˚û˚˚˚˚˚˚˚h˚˚˚˚˚д˚˚˚˚˚˚˛Ç˚˚˚˚
mudlib ˚˚˚˚˚˚˚µ˚Э˚龘˚˚.

˚˚ЩЭ˚˚:

˚˚˚д˚˚ ˚˚˚˚˚˚˚û˚˚˚˚˚˚˚˚Щ˚˚ˊ˚˚˚˚:

"class:msg_length:msg"

msg_len ˚˚ msg˚˚ ˚˙˚˚˚˚ij˚˚˚.
˚˚˚˚˚˚˚˚˚˚˚˚˚˚˚˚û˚˚˚˚˚˚˚˚˚˚˚˚Щ˚˚˚˝˚˚˚˚

ˊ˚õ˚˚µķ˚˚˚˚嶸,
ˊ˚û˚˚˚˚˚˚˚˚˚˚ˊ˚˚˚˚Щ˚˚˚˚˚˚˚˚Щ.

say˚˚˚˚ ˊ˚˚ say˚˚˙˚˚
shout˚˚˚˚ ˊ˚˚ shout˚˚˙˚˚
tell˚˚˚˚ ˊ˚˚ tell˙˚˚
emote˚˚˚˚ ˚˚˚˚˚˚˚
broadcast˚˚ ˚˚ÿh˚˚˚ˊ虧˚˚˚˚Щ
combat˚˚˚˚ ˚˚˚˚u˚˚˚˚Щ
combat_self˚˚ ˊ˚˚˚˚˚Ł˚˚˚˚˚˚˚u˚˚˚˚Щ
combat_other˚˚ ˚˚˚ ˚˚˚˚˚u˚˚˚˚Щ
combat_*˚˚ ˚˚˚˚˚ص˚˚˚u˚˚˚˚Щ
room_description ˚˚˚˚˚ص˚ij˚˚˚˚
room_name˚˚ ˚˚˚˚˙˚˚˚˚˚˚
inventory˚˚ ˚˚˚˚˚Θ˚˚ŵĶ˚˚˚
item_description ˚˚Ʒ˚˚˚˚˚
status˚˚˚˚ ˚˚˚˚˚˝˚˚Щ
status_hp˚˚ Ŀǰ˚˚˚˚˚˚

status_sp�� Ľǰ�ķ���
status_sobriety�� Ľǰ�ľ���″��
status_*�� ������″
score���� ������̆�����Ш
score_exp�� ����¨
score_money�� ��ǯ̈́̈́̈́̈́̈́̈́̈́̈́̈́̈́̈́ц̈́̈́̈́̈́̈́
developer�� ̈́̈́̈́̈́̈́̈́̈́̈́ъ̈́̈́ҷ̈́̈́̈́̈́̈́ 氈
class_fighter�� ��ő��u‘����Ш
class_mage�� ��ō�巨ъ����Ш
class_thief�� ��ő̄̈́̈́̈́̈́̈́̈́̈́Ш
class_priest�� ��ō̈́̈́̈́̈́̈́̈́̈́̈́Ш
class_*���̈́ ̈́̈́̈́̈́̈́̈́ɥő̄̈́̈́̈́̈́̂̈́Ш
race_human�� ��ő̄̈́̈́̈́̈́̈́̈́̈́̈́Ш
race_elf�� ��ō̄̈́徫̈́̈́̈́̈́̈́Ш
race_dwarf�� ��ő̄̈́厊̈́ˬ̈́̈́̈́Ш
race_*̈́̈́̈́̈́ ̈́̈́̈́̈́̈́̈́̈́̈́̈́ō̄̈́̈́̈́̈́̈́Ш

*** ̈́̈́̈́̈́̈́گ̈́̈́̈́ẅ́̈́ ***
bitmap̈́̈́̈́̈́ ̈́̈́̈́ĵ̈́̈́̈́ɕ
bitmap_*̈́̈́ ̈́ض̈́̈́ĵ̈́̈́̈́ɕ
drawing̈́̈́̈́̈́ ̈́̈́̈́Ļ̈́ɕ
drawing_*̈́̈́ ̈́ض̈́̈́̈́̈́ ̨Ļ̈́ɕ

---

̈́̈́̈́̈́: Spock @ FF 97.Aug.8.

---

[̈́ص̈́̈́̈́hx̨]

---

# MudOS 驱动程序 (MudOS driver)

---

## * MudOS 是啥 ?

---

MudOS 驱动程序 (driver) 是提供 mud 低阶支援的程序 (以 C 撰写的 LPmud). 这个程序包括以下的许多功能:

1. 0) 通过一个通信端口接受远端机器 (remote machines) 连线, 并将这些连线接上连线物件 (login object) (在 TMI 中是 /adm/login.c).

1. 1) 提供一套外部的函数 (external functions, efuns, 译按: 在原 MudOS v21c2 doc/efun 目录下所附的一大堆文件中, 称 efuns 为 emulated functions (外部函数), 哇勒....) , 可以在 LPC 物件中调用使用.

1. 2) 通过 new(文件名称) 或 clone_object(文件名称) 外部函数, 编译文件为简洁的内部表徵形式 (compact internal tokenized form)

1. 3) 解释 (interpret) [ 或执行 (execute) ] 已成为表徵形式 (tokenized form) 的物件. 执行源代码有以下两种方法:

    - a) 驱动程序以使用者的输入为准 (通过通信端口) , 调用物件中的函数. 这些特定的函数调用, 根据 mud 的物件所指定的使用者指令 (command) 和函数间的关系为准 [ 通过 add_action(函数名称, 指令名称) ] . 在 LPC 物件中, 驱动程序也会从特定的外部函数 (像 init, create, clean_up 等等)中进行函数调用 .

    - b) 物件可以通过 call_other(物件, 函数名称, 参数...) 外部函数, 让驱动程序执行其他物件中的源代码. 与 call_other() 等效的另一种写法是物件->函数名称( 参数... ).

---

翻译: Spock @ FF 97.Aug.8.

---

回到上一页

---

# ??? (objects)

---



---

h?????????ç?? (functions, ???Ï methods)
?`?????????????\ ??? (variables)? ?????.
????h??????e???????h??????,
??????????????????Í?ñh??????.

?? mud ?e??κζ????????????? (object). ???輪
?????????ÿh????Þ????
(??????????????????,
??????????????????(interactive),
??????????ώ????????) ő?????????. mud
?????ÿh?????? (???????? (virtual object) ????)
????hЩ?? LPC д???ļ? (?? mud ??Ł¼?ű??)
?????????. ??Щ?ļ?????????? mud ????,
Ç?????????????????????????????????????Ö.

---

????: Spock @ FF 97.Aug.8.

---



---

# П������ (preprocessor)

---

LPC П������ (preprocessor) ��
(93.07.17 ����)

---

�� LPC ������ (compiler) ����,
П�������Ъ������ǰ��, ���ş���
±�������:

- ����Ķ����Ъ���
- ��
- ������ (#if, #ifdef, #ifndef, #else, #elif, #endif)
- ���
- '���ص��ı
- ��  ����¸'

ǰ������ C ��. ������ч C ����,
���¸������~��ļ����  ����.

ע��:

ǰ���# ��и������ (���# ),  ## ###ű������ÿ֮ej֮h����, ǰ֮治�����пh֮.

---

## ����Ķ����Ъ���

---

#include ��ş֮'֮֮.

֮ 1 :  ### #include <file.h>

֮ 2 :  ### #include "file.h"

ע��:

֮ 1  ## ###', ###єт֮ı##### (include) Ł¼##,
Ψ## file.h. (## TMI ### /include)

• 2 ►► ►►►►‘, ►►►ڳ►►►►ı►►►►Ĵ►h►►Ł¼►►,
Ψ►►►►►►►õ► file.h

#include ►►►►►►►►►►►►►►►ķ►‘,
►►h►►►ı►►►►►►►►►h►►►ı►.
►►h►►►ı►►з►►►►► #include "file.h" ►►►►►, ►►►►
file.h ►►►►►►ˮ►ę►►► #include ►►►Ĵ►λ►►.
►►ÿ►►►►►►►►±►►►ħ,
►►►►►►►►õ►►ı►Ẋ►►►►►±►►►h►►.
►►►►►►õ►►ı►►►►►►б►►►►►►►►►►►ı►►e►►►
►►►►►►ı►ħ►►►►►►ŧ►►►►►►►Ĵ►►► (duplicate-name
error) (►►►►►►►► file.h ►►►►►►  ►►► #include ►►λ►►,
h►►►►►►►►►►►►►►).

---

►►

---

►  ►彫►►►►►Ղ►►►►►►e►hIIIJ►►►►œ►►►►►►►►►.
Ł►►►►►►►►►►ڳ►►►►س►►►þ►ڸ►►►►►►►►►►►►►►ˮ►
(constant).

• 1 :  #define identifier token_sequence

• 2 :  #define identifier(id_list) token_sequence

ˮ►►:

►►►►►►►, Ղ►►►►►►e► identifier
′►ô►дЙ►►►►к►►ó►►►►►►►►►►►►►►►

stack[0] = 0, stack[1] = 2, stack[2] = 4 ....

```
#define STACKSIZE 40
#define INITCELL(x) 2*x

int *stack;

create() {

        int i;

        stack = allocate(STACKSIZE);

        for (i = 0; i < STACKSIZE; i++)

                stack[i] = INITCELL(i);

}
```

(undefine) (compiler)

: #undef identifier

:

#undef identifier.

---

---

(flexibility). identifier ,

(#ifdef <identifier>

#ifndef <identifier>
#if <expression>
#elif <expression>
#else
#endif

ˮ��:

��<identifier> ��h���Ğ̃Ď̃̃̃̃ (��Ö̃ö̃���) ��
identifier.
��Ö̃̃̃̃̃ij̃̃̃̃̃̃̃̃Ğ̃̃̃̃̃õ̃̃̃ļ̃̃̃̃ŧ̃̃̃◂
̃̃̃Ȼ̃̃̃̃̃̃̃̃̃̃̃̃̃̃̃̃̃̃̃̃V̌̃̃̃̃ķ̃̃̃ (symbol).

̃̃̃<expression>̃̃ ̃̃h̃̃̃̃̃̃̃̃̃̃̃̃̃̃̃̃̃̃̃

```
        #else
        #�� if expression
        #�� endif

���� 1:

    /* '�� #if 0��
    ������������h��Ղ-�����м���ע�� (comment).
    ������������h�������ǰĞƟ-���
    ����  λ, ���µij���ɟ��. */

    #if 0

    /* ڲ,
    ��������ˋ���
    (��ˍ�������) 0 , ���Ղ~Ղ-���

    ����. */

        write(user_name + " ��  " + total_coins + " .\n");

    #else

    /*�� ������������ (if ����, else
    ���p���), ���Ի-����~γ���. */

        printf("%s ��  %d \n", user_name, total_coins);

    #endif

���� 2:

    //�� ������ TMI ��  /adm/simul_efun/system.c
    ��д����.

    #ifdef __VERSION
        string version() { return __VERSION__; }
    #elif defined(MUDOS_VERSION)
        string version() { return MUDOS_VERSION; }
    #else
```

```
#�� if defined(VERSION)
        string version() { return VERSION; }
#�� else
        string version() { return -1; }
#�� endif
#endif
```

---

## ����

---

#echo ������������������������ stderr (STanDard ERRor, ��������) з����Ц.
��  ������Мт���ʰ�dz�����.

**�:**

```
#echo This is a message
```

**ˮ��:**

```
#echo ���o�����Ц������, ������з��.
��Цj���ü��  ".
```

---

## ص��ı  ������

---

������������������ъ��Ĺ���.

**�:**

```
#pragma keyword
```

Lǰ���õ keyword ��������:
��

- strict_types
- save_binary
- save_types
- warnings
- optimize

- error_context

'�� #pragma no_keyword ���Ѳ-ز-ٍ-ٍ-ٍ-Ĺ���.

ˇ��:

- 'strict_types' ٍ-ٍ-ٍ-ٍ-ٍ-ٍ-ٍ-ٍ-ٍ call_other() ٍ-Í���, ٍ-ٍ-ٍ-ٍ-ٍ-ٍ-ٍ-ٍ-

Notes:

@����- ����h������ write() ����_����.

@@�- ����h����_�������, ���Ĵ�x_����
(body pager).

ن������
(end
marker)ĭ, '@@ @marker @@ @@marker.
����ĕ�������������'������'������.
������ marker
@@Ĭ����,
@@@ @ @@ @@.'@@ @,
@@@¾@@@h@@@_@@@,
@@@д@@@@@ \n.@@'@@ @@,@@
@@@¾@@@h@@@_@@@@@@@,
@@ÿh@ʒ@@@h@@@_@@@.

@@@@ 1 :

```
int help() {
	write( @ENDHELP
@@@@@@@@@@@@¡@
It's hopelessly inadequate.
ENDHELP
	);
	return 1;
}
```

@@@@@:

```
int help() {
	write( "@@@@@@@@@@@@¡@\nIt's hopelessly inadequate.\n"
	);
	return 1;
}
```

@@@@ 2 :

```
int help() {
        this_player()->more( @@ENDHELP
������������¡�
It's hopelessly inadequate.
ENDHELP
        , 1);
        return 1;
}
```

������:

```
int help() {
        this_player()->more( ({ "������������¡�", "It's
        hopelessly inadequate." }), 1);
        return 1;
}
```

---

����: Spock @ FF 97.Aug.9.

---



---

# ģ��  ���  (simul_efun)

ģ���  ����  (simulated efunctions (simul_efuns) )

mudlib ����дģ����  ����  (efunction).
����ģ����  ������
����h������������ļ���
(���ļ�������5�Ł¼λ�ö������  config.example ��) (�:
config.example Ïh��'���ļ, ����  driver
�����ĥ������ę��  mud ��ę̃��. ������  ES 2 mudlib
����, ������  es2/adm/etc Ł¼���ц  config.ES2
����ļ. ��  MudOS v21c2 '  ������, ��  MudOS_v21c2/src
Ł¼пп����ц  config.example) .

����h������ĥ, ����h�������������  (��  call_other
����)  ��������û�ж����,
Χ̨����h���  ����ĥ,
������������Ψ��ģ���  ������������,
������'������p�Ïģ���  ����.
������������д'���, ������������  call_other
ę̨ķ'���ô'���  (������ģ���  ������ļ�����).
��ш��, call_other��  ę̨ķ  ������Ç���ŧ���
(typecast), ��Ï������������Ƨ��趨������.

ģ���  ;.
����h������  ��  ������������
(����ę̨ķ), ļ����Ç  ������������
(��Ç��ģ���  ����ę̨ķ��Ï̈́ķ̈́ķ  ������).
������������  ¤ move_object() ¤  ����,
������Ƨ���ḧ́ķ̈́ķ̈́ķģ���  ����.
���Χ̨����  move_object()
��ģ���  ������������鱅������,
Ȼ���  ��  efun::move_object(). efun::
���į̆ǫ̂ḧ́ķÇ̈́ķ̈́ķ̈́ķz,
̈́ķ̈́ķ̈́ķ̈́ķ̈́ķ̈́ķ̈́ķ̈́ķ̈́ķ̈́ķ̈́ķ  ����  move_object()

---



---



---

# LPC ����

---



---

# LPC ����  ����  ű

---

## ������������eķ����į���



---

ص���hx

---

# for

## * LPC ﷺ for ﷺ :

LPC ﷺ for ﷺ C ﷺ. ﷺ ﷺ:

```
for ( ﷺ; ﷺ; ﷺ ) {
    ﷺ; ...;
}
```

ﷺ, ﷺ ﷺ 0ﷺ. ﷺ ﷺ ﷺ, ﷺ ﷺ 1ﷺ. ﷺ 1ﷺ ﷺ 0, ﷺ ﷺ. ﷺ 2ﷺ ﷺ.

ﷺ break ﷺ continue ﷺ, ﷺ ﷺ (ﷺ ﷺ 2ﷺ).

ﷺ for ﷺ, ﷺ ﷺ.

ﷺ:

```
int i;

for (i = 0; i < 10; i++) {
    write("i == " + i + "\n");
    write("10 - i == " + (10 - i) + "\n");
}
```

ﷺ: Spock @ FF 97.Aug.9.

ﷺhx

# function

---

## * LPC ?í??? (function ???Ï method) :

---

LPC ?í?????? C ?ş??????, ???????Ш?????? (??
ANSI C ?ş?????Ï????).
?  ?????:

```
????¨?????? ??????????( ?????h??????
?????h, ???????????????? ???????, ... )
{
    ?????????;
    ...;

    ????????;
    ...;

    return ?????¨?????;
}
```

??ы??,
??????¨??????????????????Þ????¨?????????.

??????????????????h???,
??????????¨?????Ï void
(?  ????¨). ????????:

```
void ??????????( ?????h??????? ?????h, ... )
{
    ???;
    ...;
}
```

???????e?????????í???:

```
??????????( ?????h, ???????, ???????, ... );
```

# if

## * if ?? else ???:

LPC ?? if ????? C ?ş????. ?  ????:

```
if (????‘)
    ???;
```

?Ĵ????:

```
if (????‘) {
    ???;
}
```

??????:

```
if (????‘h) {
    ?????h;
} else {
    ??????;
}
```

??????:

```
if (????‘h) {
    ???h;
} else if (????‘??) {
    ????‘??;
}
```

else if ?X?û??Ĵô??????є?????.

??h?????õij???ű?? ? : ?????. ???? C ?ş??h??. ?  ????:

```
������ ����‘h ? ����‘hÏ��������������‘��:
����‘hÏ��������������‘��

ijЩ������, ? : ������³���ŭ����д:

������ if (����‘h)
������������ ���= ����‘��;
������ else
������������ ����= ����‘��;

������γ������п������������:

������ ����= ����‘h ? ����‘��:����‘��;
```

---

����: Spock @ FF 97.Aug.9.

---



---

# include

LPC ?? #include ????:

?:

#include <file.h>

??h???:

#include "file.h"

?"??:

#include "file.h" ??Ľǰ??Ľ¼??Ψ?? file.h.

#include <file.h> ???  ?ǰ??ɛ??????Ľ¼ (standard system include directories) ??Ψ?? file.h (TMI mudlib ??, ????Ľ¼?? /include ?? /local/include).

??Ч C ?????Ö??ָ??, LPC ?? #include ?????? C ????. ??????ǰ???Ч??????"?, #include ??????? (include) ??????ļ????  ķ???. ?? #include "file.h" ????h???ļ???, ???ǰ? file.h ???????ō??д??????? #include "file.h" ??λ??. ÿ?????????±???ʰ, ????????õ?Χ?????? ±???h??. ??????õ??????ó??????????????????????e?ł ????ʰ?????ŀ??????? (duplicated-name error) ?Ĵ??? (???????? file.h ?e?????ō??д????????, ???h??).

????: Spock @ FF 97.Aug.9.

# inherit

LPC ֵ inherit ֵֵֵ:

ֵ:

inherit ·ֵֵֵֵֵֵ;

·ֵֵֵֵֵֵj ֵֵֵֵֵֵֵֵֵֵ " ,ֵֵ ֵֵֵֵ "/std/object".

inherit ֵֵֵֵş LPC ֵֵֵֵeֵֵֵֵֵ
(ֵֵֵֵֵֵֵֵֵֵֵֵֵֵֵֵzֵֵֵֵĹֵֵֵ).ֵֵ ֵֵֵֵ
(inheritance) ֵֵֵֵֵֵֵֵֵֵíֵֵֵֵֵֵ. ֵֵï
MudOS ֵֵֵֵֵֵֵֵֵֵֵō (global data)
ֵֵֵֵֵֵֵֵگֵ,
ֵֵֵeֵֵֵֵֵֵֵֵֵֵֵֵ(compile) ,
ֵֵֵֵֵֵֵ ֵֵֵֵֵֵֵֵֵֵֵֵüֵֵֵֵֵֵֵֵֵֵֵֵhֵαֵֵֵֵ
ÿhֵֵֵֵֵֵֵ¨ΩֵκֵōֵֵֵֵֵֵֵֵĻֵֵֵֵֵhֵֵ.
ֵֵֵֵֵֵֵֵֵֵֵֵֵֵֵֵ A ֵֵ B ֵֵֵֵֵ C , ֵֵֵ±ֵֵֵ
A ֵֵ B ֵֵֵֵ′ C Ҳֵֵֵ±ֵֵֵhֵֵ. ֵֵֵֵ,
ֵֵֵֵֵֵֵֵֵ A ֵֵ B ʯzֵֵ C ֵşֵֵōֵֵֵֵֵֵֵ , A ֵֵ B
ֵֵֵֵhֵֵ C ֵşōֵֵֵֵֵĽֵֵֵ. ֵֵֵĻֵֵֵ (update) A
ֵֵֵֵÿֵֵ B ֵֵōֵֵֵֵ ( C ֵşֵֵ) , ֵֵֵֵƇֵֵ.

ֵֵֵֵֵֵֵֵ A ֵֵֵֵֵ B. ֵֵֵ A ֵֵֵƧֵֵֵֵֵ B
ֵֵֵֵֵֵֵֵֵֵֵֵzıֵֵֵֵ ֵֵֵ. ֵֵֵ A
ֵֵֵֵֵֵֵhֵֵֵֵ B ֵֵֵֵֵíֵֵֵ, ֵֵ Aֵֵ
ֵֵֵֵíֵֵֵֵтֵƚʊ (override) B ֵíֵֵֵֵֵֵֵ. ֵֵֵ A
ֵֵ′ֵֵ B ֵĶֵֵֵ, ֵֵֵֵֵֵֵֵֵֵ: ֵֵֵֵ A
ֵֵֵֵֵֵֵhֵֵֵֵֵï query_long ֵíֵֵֵ, ֵֵ Aֵֵ
ֵֵֵֵֵֵֵֵ /std/object.cֵֵ ֵֵֵֵeֵ query_long ֵֵֵֵ, ֵֵ
A ֵֵֵֵ′ֵֵ object::query_long() ֵֵֵֵֵֵֵ. ֵֵֵ A
ֵֵֵֵֵֵֵֵֵ B ֵֵֵֵֵֵōֵֵֵֵ, ֵֵ Aֵ ֵֵɵֵֵֵ B
ֵíֵֵֵֵֵḍֵûֵֵֵֵ B ֵֵֵֵōֵֵֵֵ. ֵֵֵ B
ֵֵֵֵֵhֵֵőֵֵֵֵ, ֵֵ A ֵʙֵûֵֵֵֵֵֵ (declare) ֵֵ,
ֵֵ A ֵֵֵֵ′ֵֵֵֵֵōֵֵֵֵ, ֵֵֵֵֵֵֵ A

---

: Spock @ FF 97.Aug.10.

---

hx

---

# prototypes

---

## * ������ ⁄⁄ (prototype):

---

LPC ⁄Í⁄⁄⁄ ⁄⁄⁄⁄ ANSI C ⁄dz⁄⁄⁄⁄⁄.
⁄⁄⁄⁄ ⁄J⁄⁄⁄⁄⁄⁄Mõ⁄⁄⁄⁄є⁄⁄,
⁄⁄⁄⁄⁄⁄ĥ⁄⁄ĭ⁄⁄⁄⁄⁄⁄ ('forward' declaration).

⁄⁄⁄⁄⁄⁄¨⁄⁄⁄⁄⁄⁄⁄⁄⁄( ⁄⁄h⁄⁄⁄⁄⁄⁄⁄⁄⁄⁄⁄⁄
⁄⁄⁄⁄h, ⁄ǰ⁄⁄⁄⁄⁄⁄⁄⁄⁄⁄⁄⁄⁄⁄⁄⁄⁄⁄⁄, ... );

⁄⁄⁄⁄Í⁄⁄⁄⁄ᴗ⁄⁄⁄ ⁄⁄⁄⁄⁄,
⁄⁄⁄⁄⁄⁄⁄⁄⁄Ç⁄⁄⁄⁄⁄⁄⁄:

⁄⁄⁄⁄⁄⁄¨⁄⁄⁄⁄⁄⁄⁄⁄⁄⁄( ⁄⁄⁄⁄h, ⁄⁄⁄⁄⁄⁄⁄, ... );

---

⁄⁄⁄⁄: Spock @ FF 97.Aug.10.

---

⁄ص⁄⁄⁄hx

---

# switch

---

* switch ���

---

LPC ⓒ switch ⓒⓒⓒⓒⓒ C⓪ ⓒⓒ switch ⓒⓒ伸ⓒⓒhⓒⓒ.
Ψhⓒⓒ*ⓒ̂ɻⓒⓒⓒ LPC⓪ ⓒⓒ switchⓒⓒ
ⓒжⓒⓒⓒⓒ ⓒⓒⓒⓒⓒ‘ⓒⓒⓒⓒ'ⓒⓒⓒ_ⓒⓒⓒⓒⓒⓒⓒⓒⓒⓒ.
ⓒ ⓒⓒⓒⓒ:

```
switch (ⓒⓒⓒⓒ‘) {
        case ⓒⓒⓒⓒh : ⓒⓒⓒh;
                break;
        case ⓒⓒⓒⓒⓒⓒ :ⓒⓒ ⓒⓒⓒⓒ;
                break;
        default :ⓒⓒ ⓒⓒⓒⓒⓒ;
                break;
}
```

switch ⓒⓒⓒ٦ⓒⓒⓒhⓒⓒ if else if else if else ⓒű. ⓒⓒⓒⓒⓒ switch ⓒⓒⓒⓒ:

```
tmp = ⓒⓒⓒⓒ‘;

if (tmp == ⓒⓒⓒⓒh) {

        ⓒⓒⓒh;
        ...;
} else if (tmp == ⓒⓒⓒⓒⓒⓒ) {
        ⓒⓒⓒⓒ;
        ...;
} else {
        ⓒⓒⓒⓒⓒ;
        ...;
}
```

switchⓒⓒ ⓒⓒ if ⓒⓒⓒⓒⓒⓒⓒⓒIJⓒⓒⓒ, ⓒⓒⓒⓒij⓪ case

# while

LPC ‧‧ while ьж‧‧:

LPC ‧‧ while‧‧ ьж‧‧‧‧ C‧‧ ‧ş‧‧‧‧. ‧ ‧‧‧‧:

```
while (‧‧‧‧‘)
    ‧‧‧;
```

‧‧‧‧‧Ь‧‧‧h‧‧‧‧ {} ‧‧‧‧‧‧X‧‧‧‧‧ij‧‧‧. ‧‧‧‧‧‧‧‧‧:

```
while (‧‧‧‧‘) {
    ‧‧‧h; ‧‧‧‧;
}
```

Ç‧‧‧‧‘‧õ‧‧Ľ‧‧Ï‧‧‧‧ (non-zero),‧‧ ‧ɔ‧‧ط‧‧‧ while‧‧ ьж‧‧‧‧‧‧‧‧‧‧‧‧.
‧‧‧‧‧‧‧h‧‧ьж‧‧‧‧‧‧ğ, ‧‧‧‧‘‧õ‧‧‧, ‧T‧‧‧‧‧‧‧‧‧‧‧‧‧‧‧. ‧‧ьж‧‧‧‧‧‧‧‧‧‧ break; ‧‧‧, ‧‧‧ while‧‧ ьж‧‧‧‧ (‧‧‧‧‧‧‧‧‧‧‧δ‧e‧‧‧‧). ьж‧‧‧‧‧‧‧‧‧‧‧‧‧ continue;‧‧ ‧‧‧, ‧‧‧‧и‧‧’‧‧ьж‧‧ (‧‧‧‧‧‧‧‧‧‧‧‧‧л‧δ‧e‧‧‧‧). ‧‧

```
int test(int limit)
{
    total = 0;
    j = 0;
    while (j < limit) {
        if ((j % 2) != 0)
            ‧‧continue;
        total += j;
        j++;
    }
```

```
        return total;
    }
```



---



---



---

# LPC ???? ???? П??????

---

??????????ęķ????ļ???



---

ص???hx

---

# define

---

Ţ������������ #define��  ����������h����,
����ļ�������Ĺ�������û�Ï  ��������.
�������ˎ, ������ï�������������:

#define apples oranges

������Ղ������e�ÿ�  apple , ���  ��Ï oranges.

---


���  : Spock @ FF 97.Aug.10.


---


ص���hx


---

# include

---



#include "filename"



---



---



---

# readme

---



---


????: Spock @ FF 97.Aug.10.


---



---

# LPC ???? ???? ????

---

## ??????????eķ????ļ???



---

ص???hx

---

# array.2d

LPC ������������ , ��.
Ç������ģ���h�������������龘��.
�������~:

```
a = allocate(10);
a[0] = allocate(10);
a[1] = allocate(10);
...ԓ֊�����...
```

Œ�������������������µķ̀'����(reference) ���0 , ��0 :

```
a[0][0]
```

������������(declare)��h������h ������(������������ (type checking), ��′�� *��
���ķ̀���) ,
��������ó�����h��h ���ϵ�����.
, ������������ţ��
(mixed)�ˇ ˇ˛
��
����′������������̣���:

```
mixed a;

a = ({ ({ 1, 2, 3 }), ({ 1, 2, 3 }) });
```

������������������, a[0] ��({ 1, 2, 3 }), ��a[0][2] ��3.

������̣���:

```
mixed a;
a = ({ 0, 0, 0, 0 });�� /* ◆趨◆◆◆◆CÏ 4 */
a[0] = ({ 1, 2, 3 });
```

a[1] = ({ 1, 2, 3 });
...ϑ֏ϑϑϑϑϑ...

---

ϑϑϑϑ: John Price a.k.a. Raistlin

---

ϑϑϑϑ: Spock @ FF 97.Aug.10.

---

ϑصϑϑϑhx

---

# buffer

---

������ (buffer) ���� LPC�� ���� (array) ��ۣ�� LPC
ۣ������ۣ��.
L���DZ��گ�������λ����.
ã������(zero-terminated) ۣ ۣ۪, buffer�� ��h�����j��۪ (associated length)��.
h����۪ (bytes)�� ������,
��ÿ۪ ۣ۪h����ۣ.

```
buf[i] = x ۣ۪ x = buf[i];
sizeof(buf);
bufferp(buf);
buf[i..j];
buff = read_buffer(file_name, ...); (������ read_bytes ۣ۪)
int write_buffer(string file, int start, mixed source);
buf = buf1 + buf2;
buf += buf1;
buf = allocate_buffer(size);
```

������Щ۪ ۣ۪۪'۪۪.

socket ۪۪۪ ۣ۪۪۪۪ĜĐ۪۪۪۪ ۣ۪,
۪۪۪Ɪʊ ۣ۪۪۪ ۣ۪۪۪۪۪۪ۣ (STREAM_BINARY (3) ۪۪ DATAGRAM_BINARY (4)۪۪ ġ‘.)

---

۪۪۪۪: Spock @ FF 97.Aug.10.

---

ۣ۪ص۪۪۪hx

---

# float

---

MudOS LPC Ŀj̆ ̨ ̨ ̨

# function

h[illegible] :

MudOS [illegible]h[illegible]Ï function[illegible] ([illegible]) [illegible]ı[illegible].
[illegible]ı[illegible]Í[illegible].
[illegible]Ç[illegible]ч[illegible]h[illegible]ijЩ[illegible] [illegible]
(efuns)[illegible] [illegible]. [illegible],
[illegible]ñ[illegible] [illegible]. [illegible]h[illegible],
[illegible]h[illegible]e[illegible] [illegible].
[illegible]Ь[illegible] (non-zero) [illegible] [illegible] ⊦[illegible],
[illegible]h[illegible]μ[illegible]. [illegible]т[illegible],
[illegible]Ç[illegible]h[illegible]. [illegible],
[illegible]ú[illegible].
[illegible]h[illegible]Щ,
[illegible]ժ[illegible]言[illegible].

[illegible]Ղ[illegible] [illegible]Ï[illegible]:

```
function f = (: local_func :);
```

[illegible] f[illegible]
[illegible],
[illegible]ı[illegible]:

```
foo(f);[illegible] map_array( ({ 1, 2 }), f);
```

[illegible]ժ[illegible] f:

```
x = evaluate(f, "hi");
```

[illegible]h[illegible]ñ, [illegible] f[illegible]Í[illegible], [illegible] "hi"
[illegible]n[illegible] [illegible]. [illegible]Ч[illegible]:

```
x = local_func("hi");
```

'�ú������íô���, ��������'�ò��í���, ��Ç�ı言
��������.

ʊ�� evaluate() IJ�������Á���, ��ˇ��ز���.
������������µ
��:

```
void set_short(mixed x) { short = x; }
mixed query_short() { return evaluate(short); }
```

����, ˆ ˆ ˆ ˆ ˆ ˆ ˆ ˆ ˆ ˆ set_short("Whatever"); ϑ-��
set_short( (: short_func :) ); ��Ч��.

---

Ej˘������˙������ :

---

˜ ˆ ˆ í˙˙˙˙˙˙˙˙˙˙˙˙˙˙˙˙˙˙˙˙.
˜Æ ˆ ˆ ˆ ˆ ˆ ˆ ˆ ˆ ˆ

������Þ������ек������ (clones).
��Ҳ����'ò���:
��

```
        void create() {
                int i;
                function f = (: write, "Hello, world!\n" :);

                for (i=0; i<3; i++) { evaluate(f); }

        }
```

�ɔ�ʒ��:

```
Hello, world!
Hello, world!
Hello, world!
```

ʊ��, ��ժ��������, simul_efuns��
(ġ���  ����)
���  ������������.

---

��������'�� call_other ������, ����ǰ MudOS��
��'õÍ�������������. ����'Ï (: object, function
:).�� Ç'ò����Ļ,
Ö������������������鏈��.
�����þ���:
��

```
        void create()
        {
                string *ret;
                function f = (: this_player(), "query" :);

                ret = map( ({ "name", "short", "long" }), f );
                write(implode(ret, "\n"));

        }
```

������ʒ���ʻ this_player()->query("name"), this_player()->query("short"),
ʻʻ this_player()->query("long") ʻʻʻĽʻʻ.

Ç'h������ʻ̣ʻ̈ʻ̈ü��ʻ query("short") :

```
f = (: this_player(), ({ "query", "short" }) :)
```

ʻʻʻʻʻĠ ʻʻĽʻĵʻʻʻʻʻʻo:

```
f = (: call_other, this_player(), "query", "short"
:)ʻʻʻʻʻʻʻʻʻʻʻʻʻʻʻʻʻʻ /* hʻʻʻ ʻʻʻʻʻ̣ʻʻ, 'ʻʻ
call_other */
f = (: this_player()->query("short") :)ʻʻʻʻʻʻʻʻʻʻʻʻʻʻʻʻʻʻ //
ʻʻʻʻʻʻʻʻʻʻ; ʻʻʻʻʻʻʻʻ.
```

---

ʻʻʻʻʻʻʻʻʻʻʻʻʻʻʻʻʻʻʻ (expression) ʻʻʻʻʻʻʻ. ʻʻʻʻ (:
ʻʻʻʻʻ :).ʻʻ ʻʻhʻʻʻʻʻʻʻʻʻʻʻʻʻʻʻʻʻʻʻ

```
                case 1: y = 3;
                case 2: y = 5;
            }
            return y - 2;

        ������ };

        printf("%i %i %i\n", (*f)(1), (*f)(2), (*f)(3));

    }
```

��ʒ��: 1 3 -2

ʊ��,
(*f)
(...) ���� evaluate(f, ...) ,
������������ ��Ї�����ɥ������.
�к����ˇ�����Þ (legal)��
��д��,
������������ڳ�ˬ������.

---

ʲôʰ����ˌ (evaluate) ���� ?

---

������в������ ������ˌ������ğ��ˌ
ˌ��к���. ˇ������‘ ˇ����ˌ (functional) ����ˌ��,
ˌ����'ú����ˌ��ʰˌ�к���:

```
    (: destruct, this_player() :)������������������ /*
    ������������ˌ��ʰ, ˚ɔ˚ ˚ this_player().
    ������ڳ���ʰ��ˌ. */
    (: destruct(this_player()) :)������������������ /*
    ���ժ˚'˚ô˙˚����ˌ��ʰ, ˚Ż˚ ˚ this_player()" */
```

˚ڳ˚,
������������‘˚����ˌ����,
'˚˚h����ˌ������. ˚˚Ї˚˚˚˚˚˚˚˚ˌ˚˚ˇ˚˚,
˚˚˚˚ˌ˚˚˚˚Ť˚˚˚˚˚˚˚.
˚˚˚ǿ˚˚˚˚˚˚˚˚˚˚˚˚˚˚˚˚˚˚˚˚˚˚˚˚˚˚:

(: destruct( $(this_player) ) :)������������������ //
��������h��������.

$(whatever) ��'Ç.�� whatever, ��������. ��.д'���ћ,
�'������.'����h�����Ը���Ч��:

map_array(listeners,
(: tell_object($1, $(this_player()->query_name()) + " bows.\n") :) );

`h�h call_other , ������Çÿ����Щ����.
Ҳ�������Ac��.���:

map_array(listeners,
(: tell_object($1, $(this_player()->query_name() + " bows.\n")) :) );

ъ��, ������������, Ҳ������������:

map_array(listeners,
(: tell_object, this_player()->query_name() + " bows.\n" :) );

---

����: Spock @ FF 97.Aug.10.

---

ص���hx

---

# general

---

[illegible]
[illegible] (prototype) [illegible].

[illegible]:

```
string func(int arg);
```

[illegible]:

```
string func(int);
```

[illegible]

[illegible] (runtime) [illegible].

[illegible]

[illegible] (value) [illegible] (address). int, string, float [illegible] mapping, function, object, pointer ([illegible] * [illegible]) [illegible] (allocate) [illegible] (comparison operator) [illegible] == [illegible] array, mapping [illegible].

void (  )



mixed ()



---



varargs ()

# mappings

'��ў�� (mapping)������ 1992 September 28
MudOS ў������������ļ -�� ��: Truilkan@TMI

MudOS 0.9 ś����Ï��ў蟆.
ў�������e������ (associative arrays)�� h�� (���� Perl).
h������,
(ę)
��Ï����(index) , ������'�����. ����,
(sparse arrays), Ę,
1,000,000 ,
.
ў:

1)
2) C ļ struct.
(key) struct
(field) .

h:

```
mapping x;
```

h (initialize):

```
x = ([ h : h, : , ...]);
```

(ш: x = ([]); h)

h,
e.
Ï Þ e ' 0 ( ).

������û�r�’��h��ÿ��, ��Ï������ę��  ��ʰ,
�ɔώ�� Indexing on illegal type��
(������ę������������Y��)
�ĵ���.

�µ (�  ���, ����)
��������µķ����ҳ�h��ÿ��:

```
x[key] = value;
```

��������′��������Ψ����ÿ�� x��
�e��  ��ض��(key). ���ÿ�� x�� ��ʳ  ���,
��ÿ������������ę(value)��
ɔ��ΘÅ��Ɣ̣ę̣ę̣ę̣ę̣. ���ÿ鱔û�a������  ���,
����Ƨ(dynamically) ���(allocate)�  ����Lɒ,
�����µ (ę  ���, ����) ����ÿ����.

ÿ���e̊h̊  ̊  ̊ ̊ ̊

```
mixed *idx;
map x;

x = ([ "x" : 3, "y" : 4]);
idx = keys(x);                  /* idx == ({"x", "y"}) or ({"y", "x"}) */
```

keys() h (random)
keys()

```
idx = values(x);
```

idx ({3, 4}) ({4, 3}). values()
keys()

each()
each()
keys() values()

```
mixed *pair;

while ((pair = each(x)) != ({})) {

        write("key = " + pair[0] + "\n");
        write("value = " + pair[1] + "\n");
}
```

ÿ������Ï�� (two-dimensional)
(���Ǹ��  ¤  ��ÿ��),�� ������ LPC��
������:
��

```
mapping x, y;

x = ([]);
y = ([]);

y["a"] = "c";
x["b"] = y;
```

�� x["b"]["a"]�� == "c"

ÿ��Ҳ������ *�� (�˜�) ��������єõ�
(��ʌ�є����).

```
mapping r1, r2, a;

r1 = ([]);
r2 = ([]);

r1["driver"] = "mudlib";
r2["mudlib"] = "castle";
```

����:

```
a = r1 * r2
```

����� a["driver"] == "castle";

��Ҳ���Ъ�������ÿ�����.
����ÿ  И����Ï����ÿ������ (union).

```
a = r1 + r2
```

����� a["driver"] == "mudlib" ��  a["mudlib"] == "castle"

��+=��Ҳ��'��. ��������������:

a += ([key : value]);

���Ղ-���:

a[key] = value;

����, ������������ a[key] = value ����'Ç��ǰ��• a += ([key : value]) ę���Ç������Ч��.
��Ïǰ��ę������Ç����h���µ�ǐ����  :. ([key : value]) ���ǐ  ,  ���.

ǐ鴿��'•• - (����) ��������Ç'•• map_delete() ••.

sizeof() •  ������������ж•h••ǐ�����ж•••
  •)•••, ••••)••  ������.

write("ǐ•• x •ب•  " + sizeof(x) + " ••  •ε•\n");

•ۼ•••••••• (implementation):

MudOS ••ǐ••'•ÿ�����'•••••••••Ψ•• (extensible hash table). ������Ψ••Ĵ•Ch•••• 2•• •ij••• (power of 2).•• ••••••••Ψ•••••••ת••h•••ij•ʰ,
•••••••Ψ••ľ•••Ï•••••••ά•••••••••Ψ••Ч•

---

••••.

---

MudOS ••ǐ••  •••• Whiplash@TMI ••д.
•••e•h•••  ••••• Truilkan@TMI
••д••'•ÿ•••••'•••••••••Ψ•••••滻
•••Ķ•  •• (binary tree)••.

ǐ•••••  ű••h•••••å••• Larry Wall д•• Perl
•••••••Ъ• hash.c ģ••. Perl••
•••••••••••••• GNU Copyleft general public license.

---

••••: Spock @ FF 97.Aug.11.

---

# strings

---


[illegible] (sub ranging) - [illegible]: Grey@TMI-2


---

[illegible] (substring operation)[illegible]
[illegible], [illegible] (substring). [illegible]:

[illegible][n1..n2]

[illegible],
[illegible].
[illegible], [illegible].

[illegible][n1..n1][illegible],
[illegible] n1 [illegible] (character).
[illegible],
[illegible] (null string)[illegible]"".
[illegible],
[illegible].

[illegible]:

```
str = "abcdefg"
```

[illegible]

```
str[0..0] ==[illegible] "a" str[0..-1]
==[illegible] "abcdefg"
str[-4..-2] ==[illegible] "def"
str[-7..6] ==[illegible] "abcdefg"
str[3..2] ==[illegible] ""
```

---

[illegible]: Spock @ FF 97.Aug.11.

---

[illegible]hx

# substructures

## ◆◆ LPC Substructures

### 1. Indexing and Ranging - General Introduction

◆◆Since v20.25a6, MudOS provides a way of indexing or getting slices (which I will, following common use, call 'ranging') of strings/buffers/arrays/mappings (for mappings only indexing is available) as well as means of changing the values of data via lvalues (i.e. 'assignable values') formed by indexing/ranging.

◆◆As an example, if we set str as "abcdefg", str[0] will be 'a', str[1] 'b' etc. Similarly, the nth element of an array arr is accessed via arr[n-1], and the value corresponding to key x of mapping m, m[x]. The '<' token can be used to denote indexing from the right, i.e. str[<x] means str[strlen(str) - x] if str is a string. More generally arr[<x] means arr[sizeof(arr)-x]. (Note that sizeof(arr) is the same as strlen if arr is a string).

◆◆Indexed values are reasonable lvalues, so one could do for e.g. str[0] = 'g' to change the 1st character of str to g. Although it is possible to use ({ 1,2 })[1] as a value (which is currently optimized in MudOS to compile to 2 directly), it is not possible to use it as an lvalue. It is similarly not possible to use ([ "a" : "b" ]) ["c"] as an lvalue (Even if we did support it, it would be useless, since there is no other reference to the affected mapping). I will describe in more detail later what are the actually allowed lvalues.

◆◆Another method of obtaining a subpart of an LPC value is via ranging. An example of this is str[1..2], where for str being "abcdefg", gives "bc". In general str[n1..n2] returns a substring consisting of the (n1+1) to (n2+1)th characters of str. If n1 or n2 is negative, and the driver is compiled with OLD_RANGE_BEHAVIOR defined, then it would take the negative indices to mean counting from the end. Unlike indexing though, ranges with indexes which are out of bounds do not give an error. Instead if a maximal subrange can be found within the range requested that lies within the bounds of the indexed

value, it will be used. So for e.g., without OLD_RANGE_BEHAVIOR, str[-1..2] is the same as str[0..2]. All other out of bounds ranges will return "" instead, which corresponds to the idea that there is no (hence there is one, namely the empty one) subrange within the range provided that is within bounds. Similarly, for array elements, arr[n1..n2] represents the slice of the array with elements (n1+1) to (n2+1), unless out of bounds occur. OLD_RANGE_BEHAVIOR is only supported for buffers and strings. However, I suggest you not use it since it maybe confusing at times (i.e. in str[x..y] when x is not known at hand, it may lead to an unexpected result if x is negative). One can however, also use < in ranging to mean counting from the end. So str[<x..<y] means str[strlen(str)-x..strlen(str)-y].

/* Remark: If OLD_RANGE_BEHAVIOR is defined, then the priority of < is higher than the priority of checking if it's negative. That is, if you do str[<x..y], it will mean the same as str[strlen(str)-x..y], meaning therefore that it will check only now if strlen(str)-x is negative and if so, takes it to be from the end, leading you back to x again */

Thus far, str[<x..<y], str[<x..y], str[x..<y], str[<x..] (meaning the same as str[<x..<1]) and str[x..] (same as str[x..<1]) are supported. The same holds for arrays/buffers.

��Perhaps this might seem strange at first, but ranges also are allowed to be lvalues! The meaning of doing

```
str[x..y] = foo;�� (1)
```

is basically 'at least' doing

```
str = str[0..x-1] + foo + str[y+1..];�� (2)
```

��Here x can range from 0..sizeof(str) and y from -1 to sizeof(str) - 1. The reason for these bounds is because, if I wanted to add foo to the front,

```
str = foo + str;
```

this is essentially the same as

```
str = str[0..-1] + foo + str;
```

since str[0..-1] is just "".

/* Remark: As far as range lvalues are concerned, negative indexes do not translate into counting from the end even if���� OLD_RANGE_BEHAVIOR is defined. Perhaps this is confusing, but there is no good way of allowing for range lvalue assignments which essentially result in the addition of foo to the front as above otherwise */

Hence, that's the same as doing

str = str[0..0-1] + foo + str[0..];

or, what's the same

str = str[0..0-1] + foo + str[-1+1..];

Now if you compare this to (1) and (2) you see finally that that conforms to the prescription there if we do str[0..-1] = foo!! (Yes, those exclamation marks are not part of the code, and neither is this :))

��Similarly, I will leave it to you to verify that

str[sizeof(str)..sizeof(str)-1] = foo; (3)

would lead to str = str + foo. Now, we can use < in range lvalues as well, so (3) could have been written as

str[<0..<1] = foo; (4)

or even

str[<0..] = foo; (5)

which is more compact and faster.

/* Remark: The code for str[<0..] = foo; is generated at compile time to be identical to that for str[<0..<1] = foo; so neither should be faster than the other (in principle) at runtime, but (4) is faster than (3). */

Now we come to the part where I mentioned 'at least' above. For strings, we know that when we do x = "abc"; y = x;, y has a new copy of the string "abc".

(This isn't always done immediately in the driver, but whenever y does not have a new copy, and a change is to be made to y, then y will make a new copy of itself, hence effectively, y has a new copy in that all simple direct changes to it such as y[0] = 'g' does not change x) For buffers and arrays, however, when we do y = x, we don't get a new copy. So what happens is if we change one, we could potentially change the other. This is indeed true (as has always been) for assignments to indexed lvalues (i.e. lvalues of the form y[0]). For range lvalues (i.e. x[n1..n2]), the rule is if the change of the lvalue will not affect it's size (determined by sizeof for e.g.), i.e. essentially if
n1 and n2 are within x's bounds and the value on the right hand side has size n2 - n1 + 1, then indeed changing x affects y, otherwise it will not (i.e. if you do x[0..-1] = ({ 2,3 }). This only applies to arrays/buffers, for strings it will never affect y if we assign to a range of x.

---

## 2. More complex lvalues and applications

---

��Since arrays/mappings can contain other arrays/mappings, it is possible in principle to index them twice or more. So for e.g. if arr is ({ ({ 1,"foo",3 }),4 }) then arr[0][1] (read as the 2nd element of arr[0], which is the 1st element of arr) is "foo". If we do, for e.g. arr[0][2] = 5, then arr will be ({ ({ 1, "foo", 5 }), 4 }).

/* Remark: by the 'will be' or 'is' above, I mean technically: recursively equal. (This is just if some people are confused) */

��Similarly, arr[0][1][1] = 'g' changes arr to ({ ({ 1, "fgo", 3 }), 4 }), and arr[0][1][0..1] = "heh" (note that the right hand side can have a different length, imagine this being like taking the 1st two characters out from arr[0][1], which is currently "foo", and putting "heh" in place, resulting in "heho") gives arr as ({ ({ 1, "heho", 3 }), 4 }). You should now be able to generate more examples at your fancy.

��Now I want to discuss some simple applications.

��Some of you may know that when we are doing

```
sscanf("This is a test", "This %s %s test", x, y) (6)
```

that x and y are technically lvalues. This is since what the driver does is to parse

the original string into the format you give, and then tries to assign the results (once all of them are parsed) to the lvalues that you give (here x and y). So, now that we have more general lvalues, we may do

```
x = "simmetry";
arr = ({ 1, 2, 3, ({ "This is " }) });

sscanf("Written on a roadside: the char for 'y' has value 121\n",

       "Written on %sside: the char for 'y' has value %d\n",
       arr[3][0][<0..], x[1]);
```

(7)

will result in arr being ({ 1, 2, 3, ({ "This is a road" }) }) and x being "symmetry". (See how we have extended the string in arr[3][0] via sscanf?) The driver essentially parses the string to gives the matches "a road" and 121, it then does the assignments to the lvalues, which is how we got them as above.

��The ++,--,+= and -= operators are supported for char lvalues, i.e. lvalues obtained by indexing strings. Thus for e.g. to get an array consisting of 26 elements "test.a","test.b", .., "test.z", one might use a global var tmp as follows:

```
��mixed tmp;

��create(){

       string *arr;
       ��
       ...

       tmp = "test.`";
       arr = map_array(allocate(26), (: tmp[4]++ && tmp :));

       ...

}
```

(8)

---

---

��Finally, as a reference, I will just put here the grammar of valid lvalues accepted by the driver.

��By a basic lvalue I mean a global or local variable name, or a parameter in a functional function pointer such as $2.

��A basic lvalue is a valid lvalue, and so are indexed lvalues of basic or indexed lvalues. (Notice that I did not say indexed lvalues of just basic lvalues to allow for a[1][2]). I will generally call an lvalue obtained from a basic lvalue by indexing only indexed lvalues.

��The following lvalues are also valid at compile time:

(<basic lvalue> <assignment token> <rhs>)[a_1][a_2]...[a_n]
(<indexed lvalue> <assignment token> <rhs>)[a_1][a_2]...[a_n]
��
(9)

��/* Remark: n >= 1 here */

��assignment token is one of +=, -=, *=, &=, /=, %=, ^=, |=, =, <<=, >>=.
��
��However, because of the same reason that when we assign to a string, we obtain a new copy, (x = "foo")[2] = 'a' is invalidated at runtime. (One way to think about this is, essentially, assignment leaves the rhs as a return value, so x = "foo" returns "foo", the right hand side, which is not the same "foo" as the one in x. For arrays/buffers this is no problem because by assigning, we share the array/buffer)

�������������������� Call the lvalues in (9) complex lvalues. Then the following is also a valid lvalue:

��(<complex lvalue> <assignment token> <rhs>)[a_1][a_2]...[a_n]
��
(10)

and if we now call the above lvalues also complex lvalues, it would still be consistent, i.e. (((a[0] = b)[1] = c)[2] = d)[3] is an okay lvalue (though I wouldn't suggest using it for clarity's sake :)).

��Now, the last class of valid lvalues are range lvalues, which are denoted by ranging either a basic, indexed or complex lvalue:

- ��<basic lvalue>[n1..n2]
- ��<indexed lvalue>[n1..n2]
- ��<complex lvalue>[n1..n2]

�� plus other ranges such as [<n1..<n2] etc.
(11)

��There is maximally only one 'range' you can take to obtain a valid lvalue, i.e. arr[2..3][0..1] is not a valid lvalue (note that a naive interpretation of this syntax is one equivalent to using arr[2..3] itself)

---

## 4. Compile-time errors that occur and what they mean

---

��Here I put some notes on compile-time errors for valid lvalues, hopefully to be useful for you.

Err 1: Can't do range lvalue of range lvalue

Diagnosis: You have done 'ranging' twice, e.g. something like x[2][0..<2][1..2] isn't a valid lvalue

Err 2: Can't do indexed lvalue of range lvalue.

Diagnosis: Something like x[0..<2][3] was done.

Err 3: Illegal lvalue, a possible lvalue is (x <assign> y)[a]

Diagnosis: Something like (x = foo)[2..3] or (x = foo) was taken to be an lvalue.

Err 4: Illegal to have (x[a..b] <assign> y) to be the beginning of an lvalue

Diagnosis: You did something as described, i.e. (x[1..6] = foo)[3] is not allowed.

Err 5: Illegal lvalue

Diagnosis: Oops, we are out of luck here :) Try looking at your lvalue more carefully, and see that it obeys the rules described in section 3 above.

---

## 5. Coming attractions

---

��Perhaps a pointer type will be introduced to allow passing by reference into functions. Mappings may be multivalued and multi-indexable.

---

Author : Symmetry@Tmi-2, IdeaExchange
Last Updated : Tue Jan 10 11:02:40 EST 1995

---



---

# 基础 LPC

撰稿: Descartes of Borg
23 april 1993

## 介绍



回到上一页

# 基础 LPC

撰稿: Descartes of Borg
23 april 1993

## 简介

### 如何使用此份手册及使用的名词

最近, 在 USENET 上面, 我看到许多人寻找 LPC 的使用手册. 而且在我 mud 上的神族 (immortals) 也曾经告诉我 Nightmare 的架设文件有多好. 但是在那些文件里面, 并没有适当地解释 LPC 程式语言. 所以我决定试着撰写一份使用说明. 有些事情你必须谨记在心.

LPC 是一种非常易於学习的程式语言, 而它真正的价值也如我们在现实世界所知的一样. 我从 1991 年开始玩 mud, 并於一个月内, 在名为 Orlith 的原 Bates College MUD 中, 创造出一个不起眼的区域和乐师公会. 之後, 我搬到洛杉矶 (Los Angeles) 一年, 完全没有碰电脑或玩 mud. 在 1992 年六月, 我回到 Internet 并担任 Igor 的巫师. 在 1992 年九月, 我开始撰写 Nightmare mudlib 以符合我们的需要. 因为当时 MudOS 上并没有任何 mudlib 能让人直接拿来跑, 所以後来决定把它公开出来 (当然, 现在可不是这样 :)).

所以我只有不到一年的时间认真地撰写程式. 如同主修资讯科学的人他们的哲学, 我只想搞清楚, 要完全搞懂 LPC 程式写作, 只需要 login 到 mud 中. 在此份使用手册里, 我们假设:

有人已经教过你最基本的 UNIX 命令, 例如: ls, cs, mkdir, mv, rm 等等. 你知道如何进入你的 mud 中的文字编辑程式, 并且储存一个档案. 除此以外没有其他的要求. 如果你熟悉 C 语言, 你反而会发现 LPC 虽然很像 C, 却又不是 C. 你以前对於模组化程式设计发展 (modular programming development) 的观念还会扯你後腿. 如果你从来没听过 C 程式语言 (像我在 1991 五月那时一样) , 那你只缺基本的 C 结构, 像是程式执行的流程、逻辑运算子的规则等等东西. 所以先前学的 C 对你而言并非有利, 因为能够从 C 拿来用在 LPC 上的东西, 要学起来非常容易. 熟悉 C 跟 LPC 一点关系也没有.

这份手册分成若干章节, 表示你应该按顺序阅读它们. 先读这份简介, 再按照 chapter 那个档案里面所列的目录依序阅读. 每一章开始都有一个或两个段落, 解释你在该章节所应该了解的东西为何. 在这些简介的章节之後, 就开始针对此章节的主题作令人厌烦的详细解释. 在各章结尾, 会就你应该从此章中学到的东西下个简短的结论 (如果我写的内容能让你了解的话) . 接着, 也许会有一些跟主题有关的注解, 而你并不需要去搞清楚注解内所讲的东西.

如果你按照顺序一章章研读过来, 碰到某一章的简介说你这时应该懂某些东西, 而你却搞不懂的时候, 请寄信给我吧！很显然, 我在这里写的内容没收到该有的效果, 我得弄清楚我哪里写错, 才能把错误的地方订正过来. 如果某一章的总结说你在看完该章後应该学会什麽而你没学会, 一样寄封信给我吧. 如果你的mud 在 intermud 系统里, 就寄给 descartes@nightmare. 不然, 寄给 borg@hebron.connected.com 即可. (译按: 台湾的 intermud 系统通常寄不到美国的 mud. 而 borg@hebron.connected.com 是 1993 的地址, 已无法使用, 请改用 borg@imaginary.com )

一些手册中用的基本名词:

driver (驱动程式) -

这就是游戏的 C 程式. 它接听 socket (与其他电脑通讯)、翻译 mudlib 定义的 LPC 程式码、管理记忆体中的 mud 物件 (object)、定期试着从记忆体里面把一些没用的 mud 物件清掉、定期呼叫物件等等, 都由它负责.

mudlib (mud 函式库) -

定义 mud 世界的 LPC 程式码. driver 本身不是游戏. 它只是一个创造多人环境的程式. 就某方面来说, driver 就像是一个 LPC 的编译程式 (compiler), 而 mudlib 像是编译程式的函式库 (library) (这样比喻不太好). mudlib 定义一些基本的物件, 而创造 mud 世界的人重覆使用这些物件. 举例来说, 像是 /std/room (或 /room/room)、/std/user.c (或 /obj/player.c) 等等物件都是.

area (区域) 或 castle (城堡):

mud 创造者撰写的某些物件, 这些物件用到 LPC「继承」(inheritance) 的功能, 具有基本 mudlib 物件的特性, 并转换成玩家在游戏里面使用的

物件.

object (物件):

房间、武器、怪物、玩家、袋子等等所有的东西. 更重要的是, 每一个名字结尾是 .c 的档案都是一个物件. 每个物件有不同的用途. 像是 monster.c 和 user.c 这两个物件继承 /std/living.c 这个基础物件. 而基础物件以外的物件则有的拿来复制, 也就是在记忆体中再载入一份相同的程式码; 有的则只是载入记忆体, 而被其他的物件拿来呼叫 (reference).

native (原始) 及 compat (精简):

这两个名词与最常用的两种 driver 有关系. 原始模式的 mudlib 用於 LPMud driver 3.0 以後的版本. 精简模式指的是: 你可以拿一个 2.4.5 型式的 mudlib 配合 3.0 的 driver. 原始模式的 mudlib 指的是 MudOS、CD、LPMud 列出的原始模式 mudlib. 而精简模式的 mudlib 指的是 3.0 以前的 LPMud mudlib 和 3.* 精简模式 mudlib. 我认为 Amylaar 的 mudlib 属於精简模式.

祝你顺利 !

George Reese
(Descartes of Borg)
12 july 1993
borg@hebron.connected.com
(译按: 已改为 borg@imaginary.com)

---

译者: Spock of Final Frontier 97.Dec.21.

---

回到上一页

---

# 基础 LPC

撰稿: Descartes of Borg
第一版: 23 april 1993
第二版: 25 may 1993

## 第一章: 程式撰写环境的简介

### 1.1 UNIX 档案结构

LPMud 使用基本的 UNIX 命令及档案结构. 如果你已经了解 UNIX 的命令, 请注意 (除了几个例外) 命令无法指定选项 (options). 跟 DOS 一样, UNIX 也使用阶层式 (heirarchical) 的目录结构. 所有的次目录 (sub-directories) 都附属於根目录 ( / , root) 之下. 而每个次目录之下也可以有更多的次目录. 一个目录可以有两种表示方法:

1) 用目录的全名 (full name), 或称作绝对名称 (absolute name). 2) 使用相对名称 (relative name).

绝对名称就是从根目录一路写下来, 直到该目录的名字为止. 举例来说:

/players/descartes/obj/monster

就是根目录 (第一个 / 号) 之下的 player 目录之下的 descartes 目录的之下的 obj 目录之下的 monster 目录.

相对名称使用的是相对於其他目录的名字. 以上面的例子来说, 相对於 /players/descartes/obj, 这个目录叫作 monster; 对於 /players/descartes 来说, 这个目录叫 obj/monster; 对 /players, 同一个目录叫作 descartes/obj/monster; 最後, 对 / 来说, 此目录叫作 players/descartes/obj/monster. 你可以看出来, 绝对名称与相对名称之间的不同之处在於绝对名称总是从 / 开始. 而你如果要知道一个目录的相对名称, 就得搞清楚是相对於哪个目录.

一个目录可以包括一些次目录和档案. LPMud 只使用 mudlib 里面的文字档案. 就如同目录一样, 档案也有绝对与相对名称. 最基本的相对名称是该档案的名字. 去掉档案名字之後, 剩下的绝对名称就是路径 (path). 拿一个档案举例: /players/descartes/castle.c , 则 castle.c 是档名,

/players/descartes 则是其路径.

在其他的 mud 里, 用普通的档案列表命令列出档案时, 档名开头是 . 的档案 (像是 .plan)  是看不到的.

---

## 1.2 UNIX 命令

---

跟 UNIX 档案结构一样, LPMud 也使用许多的 UNIX 命令. 大部份的 mud 中, 使用的典型 UNIX 命令有:

pwd, cd, ls, rm, mv, cp, mkdir, rmdir, more, head, cat, ed.

如果你从来没见过 UNIX 命令, 你大概会觉得这些命令没啥意义. 好吧, 它们的确没有意义, 但是你一定用得到它们. 在我们搞清楚它们是什麽东西之前, 先来讨论目前目录 (current directory). 如果你熟悉 DOS, 那你就知道什麽是目前工作目录 (current working directory). 不管何时, 你一定在某个目录里面. 这表示, 你在 UNIX 命令里面所给的任何相对档案名称或相对目录名称, 都相对於你现在所处的那个目录. 譬如说: 如果我的目前目录是 /players/descartes, 而我输入 "ed castle.c" (ed 是编辑档案的命令), 那它就假设我指定的是 /players/descartes/castle.c 这个档案.

| | |
|---|---|
| pwd | 显示你目前所在的工作目录. |
| cd | 改变你目前的工作目录. 你可以给它相对或绝对路径名称. 如果没有指定参数 (argument),  就切换到你自己的家目录 (home directory). |
| ls | 列出一个目录里面所有的档案. 如果不指定目录, 则列出目前工作目录的所有档案. |
| rm | 删除指定的档案. |
| mv | 更改指定档案的名字. |
| cp | 复制指定的档案. |
| mkdir | 制作新的目录. |
| rmdir | 删除一个目录. 该目录里面的档案必须先全部删除才行. |
| more | 分一页一页阅读一个指定的档案, 这样你的萤幕上会一次显示一页. |
| cat | 一次就把所有的档案内容全部倒给你. |
| | |

| | |
|---|---|
| head | 显示档案的前面几行. |
| tail | 显示档案的最後几行. |
| ed | 让你能用 mud 的编辑程式编修一个档案. |

## 1.3 本章总结

UNIX 使用树状的阶层式档案结构, 而这棵树的根部叫做 / (根目录 root). 从根目录分支出去的目录, 和这些目录自己分出去的目录就叫作次目录 (sub-directory). 任何目录都可以包含档案及目录. 目录和档案都能使用以 / 开头的绝对名称, 或相对於其他目录的相对名称. 你可以使用一些典型的 UNIX 命令来使用 UNIX 的档案结构. 像是: 档案列表、显示目前工作目录、等等命令. 在你的 mud 上, 上面的那些档案都应该有详细的命令说明, 让你能搞懂那些命令到底是做些什麼的. 另外, 也该有一份 mud 编辑程式的详细说明档案. 如果你没用过 ed, 你应该详细阅读那份说明档.

译者: Spock of Final Frontier 97.Dec.23.

回到上一页

# 基础 LPC

作者: Descartes of Borg
第一版: 23 april 1993
第二版: 16 june 1993

## 第二章: LPC 程式

### 2.1 关於程式

这一章的名字取得不怎麼好, 因为没有人用 LPC 写程式. 写 LPC 程式的人写的是物件 (objects). 这两种说法有啥差别？好吧, 就我们现在的目标来说, 差别在於两者档案执行的方式不同. 当你「跑」一个程式的时候, 都是从程式中固定的地方开始执行. 换句话说, 就是所有的程式开始执行的时候, 一定有个地方写清楚要从那里开始. 另外, 程式有一个固定的终止点, 所以执行程式只要执行到该终止点, 程式就中止执行. 总之, 程式从固定的开头跑到固定的结尾. LPC 物件就不是这麼一回事.

在 mud 里面, LPC 物件只是游戏 (driver) C 程式中, 显而易见的部分. 换句话说, mud 程式在 driver 里面开始与结束执行. 但是实际上, 对於创造你玩的 mud 世界来说, driver 并没有做多少事. 反之, driver 相当依赖 LPC 码, 并需要执行物件中的程式码. 所以 LPC 物件不需要有起始点, 也不需要有固定的终止点.

就像其他的程式语言, LPC「程式」可以由一个或一个以上的档案组成. 很简单, 程式要先载入 driver 的记忆体. driver 会根据本手册所教的结构, 读取物件中一行行的程式. 有一件重要的事你要先搞清楚, 就是 LPC 物件执行时没有开头也没有终止.

### 2.2 diiver-mudlib 之间的互动

我先前提过, driver 是在主机上执行的 C 程式. 它让你连上游戏, 并执行 LPC 码. 注意, 这是 mud 程式设计的一个理论而已, 也不需要比其他的方法好. 整个 mud 游戏可以全部用 C 来写. 这样游戏的执行速度快上很多, 却让 mud 缺乏可塑性, 使巫师在游戏正在执行的时候无法加入新东西. DikuMUD 就是全部用 C 写成的. 相反的, LPMUD 的理论就是

driver 不该决定游戏内容, 而游戏内容应该决定於游戏中的个别事物, 并能够在游戏执行时加上东西. 这就是为什麽 LPMUD 使用 LPC 程式语言. 它能让你用 LPC 定义游戏内容, 交给 driver 依需要读取并执行. 况且学 LPC 要比 C 容易得多, 这样让更多人能加入创造世界的过程.

一旦你用 LPC 写了一个档案 (假设是用正确的 LPC), 它只是躺在你主机的硬碟里不动, 直到游戏中有东西参考 (reference) 它. 当游戏中有东西终於参考到它时, 这个档案就会被复制一份到记忆体里面, 并且呼叫这个物件中一个特殊的函式 (function). 呼叫这个函式的目的是初始化 (initialize) 这个物件中的变数. 现在, 别管你脑袋里才看到的上两句话, 因为一个对程式设计完全陌生的人来说, 哪里会知道函式或变数到底是啥东西. 现在重要的是要知道 driver 读取主机硬碟里面的物件档案, 复制一份之後扔进记忆体储存 (既然是复本, 也就可以有许多不同的版本). 你稍後会知道什麽是函式、什麽是变数, 并搞清楚到底游戏中的一些东西是怎麽参考你的物件的.

---

## 2.3 将一个物件载入记忆体

---

虽然一个物件里面并没有规定要从一个固定的地方开始执行程式, driver 却要先找到一个固定的地方并执行之, 才能初始化一个物件. 在精简模式的 driver 上, 这是一个叫作 reset() 的函式. 在原始模式 mud 中, 则是 create().

LPC 物件是由变数 (variable) 所组成的 (会更改的值) 而函式是处理这些变数的程式. 函式经由 LPC 语法结构来处理变数, 语法结构包括: 呼叫其他函式、使用外部定义函式 (externally defined functions, efuns)、基本的 LPC 运算式 (expression) 和流程控制 (flow control mechanism).

前面这些听起来乱七八糟的吧？让我们从变数开始着手. 拿「等级」变数来说吧, 等级可以随情形不同而改变它的数值, 而不同的事物也使用玩家的等级数字作出不同的事. 举个例: 如果你是等级十九级的玩家, 则等级变数的数值就是 19 . 如果你的 mud 是旧的 LPMud 2.4.5 系统, 等级 1 到 19 级是玩家, 20 级以上是巫师, 则会有许多事物会询问你的等级变数值, 判断你能不能使用巫师的动作. 基本上, 任何 LPC 物件就是一堆会随时间不同而改变的变数组成的. 发生在物件身上的事, 都基於该物件的各个变数里头的数值. 而常常也有许多事会更改变数.

所以无论何时, 一个 LPC 撰写的物件被其他在记忆体的物件拿来参考

时, driver 就寻找这物件里面所要找的值在哪里 (但是现在还没有任何数值) . driver 找过之後, 就呼叫物件中的 reset() 或 create() 函式 (视不同 driver 而定) , 来设定该物件一开始的变数值. 就这样, 经由「呼叫」「函式」处理变数.

虽然绝大多数的 LPC 程式码都从 create() 或 reset() 开始执行, 此处却不是 LPC 程式码开头的地方. 事实上, 没有这两个函式也没关系. 如果你的物件一开始所有的值都是 NULL (虚无) 指标 (在此, 虚无指标我们先当它是 0 吧) , 那你就不需要 create() 或 reset() 函式. 所以, 每个物件开始执行程式码的地方都可能完全不同.

现在让我们搞清楚这整章在讲些什麽. 问题是: 一个完整的 LPC 到底是由哪些东西组成的 ? 好, 一个 LPC 物件简单来说, 就是一个或一个以上的函式组合起来, 处理一个以上的变数 (或是不处理变数也行) . 各个函式之间完全不用管它们摆的先後顺序. 换句话说:

---

```
void init() { add_action("smile", "smile"); }

void create() { return; }

int smile(string str) { return 0; }
```

---

跟底下的一样:

---

```
    void create() { return; }

    int smile(string str) { return 0; }

    void init() { add_action("smile", "smile"); }
```

---

另外有个很重要的事提醒你, 下面这个物件只有:

---

```
void nonsense() {}
```

---

这样也可以, 但是这种微不足道的物件, 它大概不会与你的 mud 中的其他物件作出正确的互动关系, 因为这样的物件没有重量、看不到.....以此类推.

---

## 2.4 本章总结

---

LPC 码没有起点或终点, 因为 LPC 码是用来创造 driver 程式使用的物件, 而非单独的程式. LPC 物件包括一个或多个函式, 其间先後顺序毫无关系, 而这些函式之中, 处理多个变数 (或根本没有任何变数) . LPC 物件只是躺在主机的硬碟里面, 等着游戏中其他的物件参考它 (换言之, 它们实际上不存在) . 一旦一个物件被参考到, 它会被载入记忆体中, 并且它所有的变数都是零. 精简模式 mud 呼叫此物件的 reset() 而原始模式 mud 呼叫 create() (如果此物件有这些函式的话 ), 让这些函式来指定一些变数的初始值. 物件中的其他函式由 driver 或游戏中其他的物件使用之, 让物件之间达到互动并处理 LPC 变数.

reset() 和 create() 的说明:

只有原始模式的 mud 使用 create() (请见本手册的 Introduction 一章, 有关原始模式和精简模式的介绍). 此函式仅用来初始化刚被参考的物件.

原始模式及精简模式的 mud 都使用 reset() 函式. 在精简模式 mud 中, reset() 有两个功能. 第一, 它用来初始化刚被参考的物件. 第二, 在精简模式的 mud 中, reset() 用来重新设定物件. 也就是说, 让物件回到最初的状态. 这样可以让一个房间内的怪物重生, 或把一道门关回去.....以此类推. 原始模式的 mud 只用 reset() 作第二种功能 (就跟 reset 的意思一样).
所以在 LP 式的 mud 中有两件重要的事情让 driver 呼叫物件中的函式. 第一件事是创造物件. 此时, driver 呼叫物件中的一个函式来初始化物件的变数值. 在精简模式的 mud 里, 由 reset() 做此工作 (要加上 0 参数, 後面的章节再讨论参数是啥). 原始模式的 mud 下, 由 create() 做此工作.

第二件事是把房间重新设定回某些基本的状况. 这些基本的设定可能会与一开始的初始值不同, 也可能相同, 而你当然也不想花时间一直去重

覆做某些事 (像是重新设定一些不会更改的变数) . 精简模式的 mud 用 reset() 函式来创造和重新设定物件. 而原始模式的 mud 用 create() 创造物件, 用 reset() 重新设定物件. 但是精简模式也不会失去所有的变数值, 因为有个方法能区分是创造物件还是重新设定物件. 在精简模式要重新设定, 则 driver 传入 1 或重新设定的数字当作 reset() 的参数. 现在这个对你来说没啥意义, 但是要记住, 你在精简模式实际上是可以把两种情形区分开来的. 另外也要记住, reset() 在创造物件时传入的参数是 0, 而重新设定物件时传入非零值.

---

翻译: Spock of Final Frontier  98.Jan.16.

---

回到上一页

---

# 基础 LPC

作者: Descartes of Borg
第一版: 23 april 1993
第二版: 17 june 1993

## 第叁章: LPC 的资料型态 (data type)

### 3.1 你现在该知道的事

LPC 物件由零个或多个变数组合而成, 而这些变数由一个或多个函式组合而成. 在程式码中, 这些函式的先後顺序是无关紧要的. 当你写的 LPC 第一次被参考时, driver 把它复制一份到记忆体中. 之後, 还可藉此复制出更多相同的拷贝.

任何一份物件被载入记忆体时, 所有的变数一开始都指向「虚无值」. 精简模式 mud 的 reset() 函式与原始模式的 create() 函式都都用於指定物件的初始变数值. 物件载入记忆体之後, 会立刻呼叫创造的函式. 不过, 如果你读这份课本之前没有写过程式, 你大概不知道什麽是函式 (function) , 或函式是怎麽被呼叫的. 就算你以前写过程式, 你大概也想知道新创造的物件中, 函式之间互相呼叫对方的过程是什麽. 回答以上这些问题以前, 你得多了解函式在处理什麽. 所以你应该先彻底了解 LPC 资料型态背後的观念. 说实在的, 在这份手册里头最无聊的主题, 也是最重要的主题, 90% 以上就是用错 LPC 资料型态 (放错 {} 和 () 不算在内). 所以说, 你得要耐心看完非常重要的这一章, 因为我觉得你如果搞懂这一章, 可以让你以後写程式大大轻松不少.

### 3.2 与电脑沟通

你应该已经知道电脑不懂人类所使用的单字与数字. 电脑所说的「语言」由 0 与 1 的「字母」所组合而成. 当然, 你知道电脑不懂人类的自然语言. 但是实际上, 它们也不懂我们写给它们的电脑语言. 像是 BASIC、C、C++、Pascal 等等, 这些电脑语言全都是过渡语言. 这些电脑语言让你能把想法组织起来, 让思考更易转换成电脑的 0 与 1 语言.

转换有两个方法: 编译 (compilation) 和直译 (interpretation) . 这两个方

法的差别在於程式语言转换成真正电脑语言的时候. 对编译的程式语言来说, 程式设计者撰写程式码之後, 使用编译程式 (compiler) 把程式码转换成电脑真正的语言. 程式在执行之前就已经转换完毕. 而直译的程式语言, 在程式执行的时候才开始转换. 因此直译的程式语言所写的程式执行起来要比编译的慢上许多.

总而言之, 不管你用什麽程式语言撰写程式, 最後都要转变成 0 与 1 才能让电脑搞懂. 但是你储存在记忆体中的变数并不是单纯的 0 与 1. 所以你用的程式语言要有个方法告诉电脑, 这些 0 和 1 到底要当作十进位数字、字元 (characters)、字串 (string)、还是当作其他的东西看待. 你可以靠着指定资料型态来办到.

举例来说, 假设你有个变数叫做 x , 而你给它一个十进位的值 —— 65. 在 LPC 里面, 你会写出下面的叙述:

---

```
x = 65;
```

---

你等一下再做像这样的事:

---

```
write(x+"\n");        /* \n 符号代表在此换行 (carriage return) */ y = x + 5;
```

---

第一行让你送出 65 和换行到某个人的萤幕上. 第二行让你把 y 设定为 70. 问题是你告诉电脑 x = 65; 时, 它不知道 65 到底是啥意思. 你认为是 65, 对电脑来说也许认为是: 00000000000000000000000001000001 而且, 对电脑来说, A 这个字母就是: 00000000000000000000000001000001 所以, 不管你什麽时候告诉电脑 write(x+"\n");, 电脑总要有个方法知道你想看到 65 而不是 A.

电脑能透过资料型态了解 65 与 A 的不同. 资料型态只是说记忆体位置中储存的指定变数到底是属於什麽型态的资料. 所以说, 每一个 LPC 变数都有变数型态指导如何转换资料. 在上面的范例里, 你应该会在程

式码「之前」加上以下这行:

```
int x;
```

这一行告诉 driver 无论 x 指向何处, 都当作「int」资料型态来使用. int 是整数 (interger, 或称 whole number) 的缩写. 现在我们已经初步介绍为什麽要有资料型态. 这样一来, driver 才能搞清楚电脑储存在记忆体中的 0 与 1 到底是代表什麽意义.

---

## 3.3 LPC 的资料型态

---

所有的 LPMud driver 都有以下的资料型态:

void (无), status (状况), int (整数), string (字串), object (物件), int * (整数指标), string * (字串指标), object * (物件指标), mixed * (混合指标)

很多种 driver (不是全部) 有下列资料型态值得讨论:

float (浮点数), mapping (映射), float * (浮点数指标), mapping * (映射指标)

少数 driver 有下列罕用的资料型态, 并不值得讨论:

function (函式), enum, struct (结构), char (字元)

(译注: 目前台湾绝大多数的 LPMud 所使用的 driver 是 MudOS, 其资料型态有些许不同之处. 请详见参考译者所翻译之 MudOS 参考文件)

---

## 3.4 简单的资料型态

---

这份简介性质的课本会介绍 void, status, int, float, string, object, mixed 这几种资料型态. 你可以在中阶课本 (intermediate book, 译注: 本作者另外有写一份中阶 LPC 手册, 译者亦有翻译) 找到像是 mapping (映射) 或 array (阵列) 这种更复杂的资料型态. 本章先介绍两种最简单的资料

型态 (以 LPC 程式设计者的观点来看) —— 整数 (int) 和字串 (string).

int 表示任何整数. 所以 1, 42, -17, 0, -10000023 都是整数 (int) 型态.
string 是一个以上的字元或数字. 所以 "a", "we are borg", "42", "This is a string" 都是字串. 请注意, 字串前後都要加上双引号 "" , driver 才能分辨 int 42 和 string "42". 也才能区别变数名称 (像是 x ) 与字串 (像是 "x" ).

当你在程式码中使用变数, 你一开始要让 driver 知道这个变数所指的是哪种变数型态. 这种处理方式叫做「宣告」(declaration). 你得在函式一开始的地方宣告, 或是在物件程式码的开头之处 (在函式之外, 任何函式用到该变数之前).
要宣告变数型态的话, 只要像底下一样, 把变数型态摆在变数的名字前便即可.

---

```
void add_two_and_two() {
    int x;
    int y;

    x = 2;
    y = x + x;
}
```

---

像这样, 这是一个完整的函式. 函式的名称是 add_two_and_two(). 函式一开始宣告一个整数变数 x, 之後宣告一个整数变数 y. 所以, 在这里 driver 有两个变数指向 NULL (虚无) 值, 而这两个变数期待的变数值是整数型态.

关於虚无 (void) 和状态 (status) 资料型态:

无 (void) 是一种很普遍的资料型态, 它不指向任何东西. 它并不是用在变数上面的型态, 而是用於函式. 你稍後会了解这里所说的事. 而现在, 你只需要知道 void 不指向任何值.

状况 (status) 资料型态是布林 (boolean) 资料型态. 就是说, 它的值是 0 或 1. 这种值常常称为真 (true) 或伪 (false).

## 3.5 本章总结

对变数来说, driver 需要知道电脑储存在记忆体中的 0 与 1 要如何转换成你想使用的形式. 最简单的 LPC 资料型态是 void, status, int, string. 变数不使用 void 的资料型态, 但是这种资料型态用於函式. 另外, 资料型态用於转换格式, 决定 driver 应该使用哪种规则处理运算, 像是 +, -.....以此类推. 举例说, 运算式 (expression) 5+5, driver 知道 5 加上 5 的值是 10. 对字串来说, 对字串使用整数加法没有意义. 所以, "a"+"b" 把 "b" 加在 "a" 的後面, 最後得出 "ab". 当你试着把 "5"+5 就会产生错误. 因为把整数加上字串是无意义的, 所以 driver 会把第二个 5 转换成 "5" 再加起来. 最後的结果是 "55". 如果你想看的结果是 10 , 你最後只得到错误的程式码. 请记住, 大多数的情况下, driver 不会像前面这样产生 "55" 这种有用的结果. 它会产生 "55" 是因为它早有一条规则处理整数加上字串的情况, 也就是把整数当成字串看待. 在大多数的状况中, 如果你在运算式或函式中使用资料型态并没有事先定义 (像是你试着把 "this is" 除以 "nonsense", "this is" / "nonsense") , driver 会呕吐并回报错误给你.

翻译: Spock of Final Frontier 98.Jan.22.

回上一页

# 基础 LPC

作者: Descartes of Borg
第一版: 23 april 1993
第二版: 22 june 1993

## 第四章: 函式 (functions)

### 4.1 回顾

现在, 你应该了解 LPC 物件由许多处理变数的函式所组成. 函式执行时就处理变数, 而经由「呼叫」执行这些函式. 在一个档案里, 函式之间的前後顺序是无关紧要的. 变数在函式里面被处理, 变数储存在电脑的记忆体中, 而电脑把它们当作 0 与 1 来处理. 利用定义资料型态这种方法, 这些 0 与 1 被转换成可使用的输出及输入结果. 字串 (string) 资料型态告诉 driver , 让你看到或你输入的资料应该是许多字元及数字的形式. 整数 (int) 型态的变数对你来说就是整数值. 状况 (status) 型态对你来说就是 1 或 0. 无 (void) 资料型态对你或对机器而言都没有值, 并不是用於变数上的资料型态.

### 4.2 什麼是函式 ?

就像数学函式, LPC 函式获得输入值, 然後传回输出值. 像 Pascal 语言把程序(procedure) 和函式 (function) 区分开来. 但是 LPC 不这样做, 而知道这种区分也是有用的. Pascal 称为程序的东西, 在 LPC 就是无传回值 (void) 型态的函式. 也就是说, 程序或无传回值函式没有传回输出值. Pascal 称为函式的东西, 就是有传回输出值的. 在 LPC 里, 最短的正确函式是:

---

```
void do_nothing() { }
```

---

这个函式不接受输入, 没有任何指令, 也不传回任何值.

要写出正确的 LPC 函式有叁个部分:

1) 宣告 (declaration)
2) 定义 (definition)
3) 呼叫 (call)

就像变数一样, 函式也要宣告. 这样一来, 让 driver 知道: 1) 函式输出的资料是什麽型态 2) 有多少个输入的资料以及它们的型态为何. 比较普通的讲法称这些输入为参数 (parameter). 所以, 宣告一个函式的格式如下:

```
传回值型态 函式名称 (参数 1, 参数 2, ..., 参数 N);
```

底下宣告一个 drink_water() 的函式, 它接受一个字串输入, 而输出一个整数:

---

```
int drink_water(string str);
```

---

str 是输入的变数名称, 会用於函式之中.

函式定义是描述函式实际上如何处理输入值的程式码.

呼叫则是其他函式之中, 呼叫并执行此函式的地方. 对 write_vals() 和 add() 两个函式来说, 你可能会有这些程式码:

---

```
/* 首先, 是函式宣告. 它们通常出现在物件码的开头.
*/
void write_vals();
int add(int x, int y);

/* 接着是定义 write_vals() 函式. 我们假设这函式将会在物件以外被呼叫.
*/
void write_vals() {
```

```
    int x;

    /*  现在我们指定 x  为呼叫 add()  的输出值. */
    x = add(2, 2);
    write(x+"\n");

}

/*  最後, 定义 add() */
int add(int x, int y) {

    return (x + y);

}
```

---

请记得, 哪一个函式定义在前都没有关系. 这是因为函式并不是由前往後连续执行的. 函式只有被呼叫时才会执行. 唯一的要求是, 一个函式的宣告必须出现在函式的定义之前, 而且也必须在任何函式定义呼叫它之前.

---

## 4.3 外部函式 (efuns)

---

也许你已经听过有人提过外部函式. 它们是外部定义的函式. 跟名称一样, 它们由 mud driver 所定义. 如果你已经撰写 LPC 程式码很久, 你大概已经发现你听到的一些式子, 像是 this_player(), write(), say(), this_object()... 等等, 看起来很像函式. 这是因为它们是外部函式. 外部函式的价值在於它们比LPC 函式要快得多, 因为它们早已经以电脑了解的二进位格式存在着.

在前面的 write_vals() 函式里, 呼叫了两个函式. 第一个是 add() 函式, 是你宣告及定义的函式. 第二个, 则是称做 write() 的外部函式. driver 早就帮你宣告并定义这个函式. 你只需要呼叫它.

创造外部函式是为了处理普通的、每天都用得到的函式呼叫、处理 internet socket 的输出与输入、其他用 LPC 难以处理的事. 它们是在 game driver 内以 C 写成的, 并与 driver 一起编译在 mud 开始之前, 让它们执行起来快得多. 但是对你来说, 外部函式呼叫就像对你的函式呼叫

一样. 不过, 任何外部函式还是要知道两件重要的事: 1) 它的传回值是什麽, 2) 它要什麽参数.

外部函式的详细资料, 像是输入参数和传回值, 常常可以在你的 mud 中的 /doc/efun 目录找到. 我没有办法在这里详细介绍外部函式, 因为每种 driver 的外部函式都不相同. 但是, 你常常可以藉由「man」或「help」指令 (视 mudlib 而定) 找到详细的资料. 例如指令「man write」会给你 write 外部函式的详细资料. 如果都不行, 「more /doc/efun/write」也可以.

看过 write 的详细资料之後, 你应该找到 write 是宣告成这样:

---

```
void write(string);
```

---

这样告诉你, 要正确呼叫 write 不应该期待它有传回值, 而且要传入一个字串型态的参数.

## 4.4 定义你自己的函式

虽然在档案中, 你的函式次序谁先谁後都没有关系, 但是定义一个函式的程式码的先後顺序就非常重要. 当一个函式被呼叫时, 函式定义中的程式码按照出现的先後顺序执行. 先前的 write_vals() 中, 这个指令:

---

```
x = add(2, 2);
```

---

如果你想看到 write() 使用正确的 x 值, 就必须把它放在 write() 呼叫之前.

当函式要传回一个值时, 由「return」指令之後跟着与函式相同资料型态的值所完成. 在先前的 add() 之中, 指令「return (x+y);」把 (x+y) 的值传回给write_vals() 并指定给 x. 在更普通的层次上来说, 「return」停

止执行函式, 并传回程式码执行的结果给呼叫此函式的函式. 另外, 它将跟在它後面任何式子的值传回呼叫的函式. 要停止执行失去控制的无传回值函式, 使用 return; 而後面不用加上任何东西. 请再次记得, 使用「return」传回任何式子的资料型
态「必须」与函式本身的资料型态相符合.

---

## 4.5 本章总结

---

定义 LPC 物件的档案是由函式所组成的. 函式依次由叁个部分组成:

1) 宣告
2) 定义
3) 呼叫

函式宣告通常出现在档案的最前面, 在任何定义之前. 不过函式只要求在函式定义之前以及任何函式呼叫它之前宣告它. 函式定义可以任何顺序出现在档案里, 只要它们都放在宣告之後. 另外, 你不可以再一个函式里面定义另一个函式.
函式呼叫则出现在其他任何函式中, 任何程式码想执行你的函式的地方. 呼叫也可以出现在自己的函式定义中, 但是这种做法并不建议给新手去做, 因为它很容易变成无穷回圈.

函式定义依序由底下的部分所组成:

1) 函式传回值型态
2) 函式名称
3) 一个左小括号 ( 接着列出参数再加上一个右小括号 )
4) 一个左大括号 { 指示 driver 从这里开始执行
5) 宣告只用在这个函式中的任何变数
6) 指令、式子、视需要呼叫其他函式
7) 一个右大括号 } 描述函式码在此结束. 对於无传回值函式来说, 如果在此还没有碰到「return」指令 (只适用於无传回值函式) , 会如同有碰到「return」指令一样回到原来呼叫的函式执行.

最短的函式是:

---

```
void do_nothing() {}
```

---

因为这个函式不接受任何输入, 不做任何事, 也不传回任何输出.

任何无传回值型态以外的函式「必须」传回一个与函式资料型态相同的值.

每一种 driver 都有一套早已经帮你定义好的函式, 它们叫做外部函式. 你不需要宣告或定义它们, 因为它们早已经帮你做好这些事. 更深入一点, 执行这些函式比起执行你的函式要快得多, 因为外部函式是 driver 的一部份. 再者, 每一个 mudlib 都有特殊函式像是外部函式一样, 早已经为你宣告并定义好. 但是不同的是, 它们用 LPC 定义在 mudlib 里面. 它们叫做模拟外部函式 (simul_efuns, 或 simulated efuns). 在大多数的 mud 里, 你可以在 /doc/efun 目录底下找到关於它们的详细资料. 另外, 很多 mud 有称作「man 」或「help」的命令, 让你可以方便地叫出这些资料档案.

程式风格的注解:

有些 driver 可能不会要求你宣告函式, 有些不会要求你指定函式的传回值型态. 无论如何, 底下有两个理由劝你不要省略以上这些动作:

- 1) 对其他人来说 (还有你自己过了一段时间之後) , 会比较容易读懂你的程式码并了解程式码的意义. 这对除错时特别有用, 有很多错误 (除了放错地方的各种括号) 发生在资料型态上 (有没有碰过「Bad arg 1 to foo() line 32」? (程式第叁十二行, 呼叫 foo() 时的第二个参数有错) ).
- 2) 大家认为这样子写程式是个好习惯.

---

翻译: Spock of Final Frontier 98.Jan.25.

---

回上一页

---

# 基础 LPC

作者: Descartes of Borg
第一版: 23 april 1993
第二版: 01 july 1993

## 第五章: 基础的继承 (inheritance)

### 5.1 回顾

你现在应该了解函式基本的功能. 你应该可以宣告并呼叫一个函式. 另外, 你应该能认识函式定义, 虽然你可能是第一次接触 LPC. 你现在并不见得能定义你自己的函式. 函式是 LPC 物件的基石. 函式中的程式码, 要别的函式呼叫它们的时候才会执行. 呼叫一个函式时, 作出呼叫的函式要给它输入值, 才能执行被呼叫的函式. 被呼叫的函式执行其程式码, 并传回某种资料型态的传回值给呼叫它的函式. 没有传回值的函式属於无传回值 (void) 型态.

仔细看过你自己的工作室程式码之後, 它看起来大概像这样 (视 mudlib 而定):

---

```
inherit "/std/room";

void create() {     ::create();
    set_property("light", 2);
    set_property("indoors", 1);
    set("short", "Descartes 的工作室");
    set("long", "此处是 Descartes 工作的地方.\n这里是一个立方
体.\n");
    set_exits( ({ "/d/standard/square" }), ({ "square" }) );
}
```

---

如果你到目前为止, 所有的课本内容都了解的话, 你应该能认出以下的程式码:

- 1) create() 是函式的定义. (嘿！ 他没有宣告它)
- 2) 它呼叫 set_property() 、 set()、 set_exits(), 没有一个函式在这段程式码中曾有宣告或定义.
- 3) 最上面有一行, 不是宣告变数或函式, 也不是函式定义！

这一章会找出这些问题的解答, 你现在应该脑中应该有这些问题:

- 1) 为什麽没有宣告 create() ?
- 2) 为什麽 set_property() 、 set() 、 set_exits() 已经宣告并定义过了 ?
- 3) 档案最上面那一行到底是啥东西 ?

---

## 5.2 物件导向程式设计 (object oriented programming, OOP)

---

继承 (inheritance) 是定义真正物件导向程式设计的特性之一. 它让你创造通用的程式码, 能以多种用途用於许多不同的程式中. 一个 mudlib 所作的, 就是创造这些通用的档案 (物件) , 让你用来制造特定物件.

如果你必须把定义前面工作室全部所需要的程式码写出来, 你大概必须要写 1000 行程式码才能得到一个房间所有的功能. 当然, 那根本是浪费磁碟空间. 再者, 这种程式码与玩家和其他房间的互动性很差, 因为每一个创造者都写出自己的函式以作出一个房间的功能. 所以, 你可能使用 query_long() 写出房间的长叙述, 其他的巫师可能使用 long() . 这就是 mudlib 彼此不相容最主要的原因, 因为它们使用不同的物件互动协定.

OOP 克服了这些问题. 前面的工作室中, 你继承已经定义在 "/std/room.c" 档案中的函式. 它拥有普通房间所需要的全部函式定义其中. 当你要制造一个特定的房间, 你拿这个房间档案中定义好的通用函式功能, 并加上你自己的函式 create() 以制造一个独特的房间.

---

## 5.3 继承如何作用

---

你现在大概猜得出来, 这一行:

---

```
inherit "/std/room";
```

---

让你继承 "std/room.c" 的函式功能. 藉由继承函式功能, 它代表你可以使用 "/std/room.c" 里面已经宣告并定义好的函式. 在 Nightmare Mudlib 中, "/std/room.c" 里面有许多函式, 其中有 set_property()、set()、set_exits() 函式, 都已经宣告并定义过. 在你的 creat() 函式里, 你呼叫那些函式来设定你房间一开始的值. 这些值让你的房间不同於别的房间, 却保留与记忆体中其他房间互动的能力.

实际的写作中, 每一个 mudlib 都不同, 所以要你使用不同一套的标准函式来达到相同的功能. 说明有哪些函式存在和它们是作什麼用的, 已经超出了这本课本的范围. 如果你的 mudlib 有自己详细的说明资料, 你会找到教你如何使用各种继承档案的说明文件以创造物件. 这些说明应该会告诉你有哪些函式、它们需要哪些输入、它们输出的资料型态、以及它们的功能.

---

## 5.4 本章总结

---

本章距离完整解释继承如此复杂的主题还有一大段距离. 本文的目的只是让你能了解如何使用继承来创造你的物件. 以後的课本将对此会有完整的讨论. 现在你
应该已经了解底下几点:

- 1) 每一个 mudlib 都有一套通用物件库, 有它们自己的通用函式. 创造者透过继承使用它们, 让撰写物件程式码这件工作更轻松, 并与其他物件之间能良好互动.
- 2) 可被继承的档案里头的函式, 每个 mudlib 都不一样. 你的 mud 里应该有说明文件解释如何使用这些可被继承的档案. 如果你还不知道有哪些函式可用, 那你就没有办法用它们. 任何时候, 都请你特别注意输入和输出的资料型态.
- 3) 你藉由底下这行继承函式的功能:

```
-----

    inherit "filename";

-----
```

filename 是被继承的物件档案名称. 这行放在你程式码的开头.

注解:

你可能看到有几处地方有 ::create() 或 ::init() 或 ::reset() 语法. 你现在不需要完全了解这个, 但是应该告诉你一点线索, 知道它到底是什麽. 「::」运算子是一种特殊的方法来呼叫继承物件的函式 (叫做范围解析运算子 scope resolution operator). 例如, 大多数 mud 的 room.c 都有叫做 create() 的函式. 当你继承 room.c 并设定 create() 时, 你所作的事称为超越 (override) room.c 的 create() 函式. 这表示不管任何东西呼叫你房间的 create() , 它会呼叫「你的」版本, 而不是 room.c 里面的那一个. :: 运算子让你能呼叫 room.c 里的 create() 而不是你的 create().

一个例子:

---

```
#1

inherit "/std/room";

void create() { create(); }
```

---

---

```
#2

inherit "/std/room";

void create() { ::create(); }
```

---

第一个例子是个恐怖的例子. 当它被载入时, driver 呼叫 create() , 之後 create() 再呼叫 create(), create() 又呼叫 create(), 这时 create() 又呼叫 create()......换句话说, 所有的 create() 就一直呼叫自己直到 driver 侦测到太深的递回 (recursion) 并跳出来.

第二个例子基本上只是浪费记忆体, 它的功能跟 room.c 没有两样. 对它

而言, driver 先呼叫它的 room.c , 然後呼叫 ::create() , 也就是 room.c 里的 create() . 其他的地方就跟 room.c 的功能一样.

---

译者: Spock of Final Frontier  98.Jan.25.

---

回上一页

---

# 基础 LPC

作者: Descartes of Borg
第一版: 23 april 1993
第二版: july 5 1993

## 第六章: 变数 (variable) 处理

### 6.1 回顾

现在你应该能利用你 mud 的标准物件库, 撰写一些简单的物件. 继承能让你使用那些物件中已经定义好的函式, 而不用自己去定义. 另外, 你应该知道如何宣告你自己的函式. 这一章将教你 LPC 的基本元素, 让你能藉由处理变数来定义你自己的函式.

### 6.2 数值与物件

基本上, mud 里头的物件都不一样的原因有两个:

- 1) 有的物件拥有不同的函式
- 2) 所有的物件都有不同的数值

现在, 所有的玩家物件都有同样的函式. 它们不一样的地方在於它们自己所拥有的数值不同. 举例来说, 名字叫做「Forlock」的玩家跟「Descartes」「至少」他们各自的 true_name 变数值不同, 一个是 "descartes", 另一个是 "forlock".

所以, 游戏中的改变伴随着游戏中物件值的改变. 函式名称就是用来处理变数的过程名称. 例如说, create() 函式就是特别用来初始化一个物件的过程. 函式之中, 有些特别的事称为指令. 指令就是负责处理变数的.

### 6.3 区域 (local) 和全域 (global) 变数

跟大多数程式设计语言的变数一样, LPC 变数可以宣告为一个特定函

式的「区域」变数, 或是所有函式可以使用的「全域」变数. 区域变数宣告在使用它们的函式之内. 其他函式并不知道它们存在, 因为这些值只有在那个函式执行时才储存在记忆体中. 物件码宣告全域变数之後, 则让後面所有的函式都能使用它. 因为只要物件存在, 全域变数就会占据记忆体. 你只有在整个物件中都需要某个值的时候, 才要用全域变数. 看看下面两段程式码:

---

```
int x;

int query_x() { return x; }

void set_x(int y) { x = y; }
```

---

---

```
void set_x(int y) {     int x;

   x = y;
   write("x 设定为 "+x+" 并且会消失无踪.\n");
}
```

---

第一个例子里, x 宣告在所有的函式之外, 所以在 x 宣告之後的所有函式都能使用它. x 在此是全域变数.

第二个例子中, x 宣告在 set_x() 函式里. 它只有在 set_x() 执行的时候存在. 之後, 它会消失. 在此, x 是区域变数.

---

## 6.4 处理变数的值

---

给 driver 的指令 (instruction) 用来处理变数值. 一个指令的范例是:

---

```
x = 5;
```

---

上面的指令很清楚. 它把 5 这个数值指定给 x 变数. 不过, 这个指令牵涉到一些对普通指令来说很重要的观念. 第一个观念是运算式 (expression). 一个运算式就是有值的一系列符号. 在上面的指令中, 运算式 5 的值指定给变数 x. 常数 (constant) 是最简单的运算式. 一个常数就是不变的值, 像是整数 5 或是字串 "hello". 最後一个观念就是运算子 (operator). 在上面的例子中, 使用了 = 这个指定运算 (assignment operator).

在 LPC 有更多其他的运算子, 还有更复杂的运算式. 如果我们进入一个更复杂的层次, 我们得到:

---

```
y = 5;
x = y +2;
```

---

第一个指令使用指定运算子以指定常数运算式 5 的值给变数 y. 第二个指令把 (y+2) 的值以加法运算子把 y 和常数运算式 2 加起来, 再用指定运算子指定给 x. 听起来一点意义都没有吧 ?

换另一种方法来讲, 使用多个运算子可以组成复杂的运算式. 在前面的范例中, 一个指令 x = y + 2; 里面含有两个运算式:

- 1) 运算式 y+2
- 2) 运算式 x = y + 2

前面曾提过, 所有的运算是都有其值. 运算式 y+2 的值是 y 和 2 的总和 (在此是 7) ; 运算式 x = y + 2 「也」有其值 —— 7. 所以运算子有两个重要的工作:

- 1) 它们「可以」像函式一样当作输入.
- 2) 它们运算起来就像本身有值一样.

现在, 不是所有的运算子的功能都像 1) 一样. = 运算子将它右边的值指

定给 x. 但是 + 就没有这种功能. 而且, 它们两个也有自己的值.

## 6.5 复杂的运算式

前面你大概已经注意到, 运算式 x = 5 「本身」也有个值是 5. 实际上, 因为 LPC 运算子如同运算式一样也有自己的值, 它们能让你写出一些非常难解、看起来毫无意义的东西, 像是:

```
i = ( (x=sizeof(tmp=users())) ? --x :
sizeof(tmp=children("/std/monster"))-1)
```

基本上只是说:

把外部函式 users() 传回的阵列指定给 tmp, 然後把此阵列元素的数目指定给 x. 如果指定给 x 的运算式值为真 (不是 0) , 就指定 x 为 1 并指定 i 的值为 x-1 的值. 如果 x 为伪, 则设定 tmp 为外部函式 children() 传回的阵列, 并指定 i 为阵列 tmp 的元素数目再减 1. 你曾经用过以上的叙述吗 ? 我很怀疑. 不过你可能看过或使用与它相似的运算式, 因为一次合并这麽多的东西在一行里面, 能提升你程式码的执行速度. 比较常使用 LPC 运算子这种特性的写法大概像这样:

```
x = sizeof(tmp = users());
while(i--) write((string)tmp[i]->query_name()+"\n");
```

取代这样子的写法:

```
tmp = users();
x = sizeof(tmp);
for(i=0; i<x; i++) write((string)tmp[i]->query_name()+"\n");
```

像是 for()、while() 、阵列.....等等东西稍後会解释.
不过第一段程式码比较简洁, 执行起来也比较快.

附注: 在本章总结之後会对所有的 LPC 运算子有更详细的说明.

## 6.6 本章总结

你目前知道如何宣告变数, 并了解宣告、使用全域和区域变数之间的不同. 一旦你熟悉你 driver 的外部函式, 你就能用许多不同的方法显示那些值. 另外, 藉由 LPC 运算子, 你知道怎麼改变并运算变数里头的值. 这当然对你很有用, 因为它让你能做一些事, 像是算出从树上摘下了多少颗苹果, 一旦苹果都摘完了, 就没有人有苹果可摘. 很不幸, 你现在只会写寥寥几行能执行的程式. 换句话说, 到下一章以前先别管苹果的问题, 因为你还不知道如何检查全部摘下的苹果数目和树上原先的苹果数目是否相等. 你也不知道特殊的函式 init(), 能让你给玩家使用新的指令. 但是你已经准备好撰写良好而复杂的区域程式码.

---

## 6.7 LPC 运算子

---

这一段将详细列出比较简单的 LPC 运算子, 包括对它们使用的值所作的事 (如果有值的话), 以及它们自己拥有的值.

在此说明的运算子有:

```
=   +   -   *   /   %   +=   -=   *=   /=   %=   --   ++   ==   !=
>   <   >=   <=   !   &&   ||   ->   ? :
```

下面, 这些运算子将全部用相当简单的方式说明之, 但是你最好把每个运算子至少都看过一次, 因为有些运算子的功能「不见得」如你所想的一样. 不过, 这段说明可以当作相当好的一个参考.

---

### = 指定运算子 (assignment operator):

| | |
|---|---|
| 范例 | x = 5 |
| 值 | 在完成它的功能之後, 「左边」的变数值 |
| 说明 | 把它「右边」任何运算式的值指定给它「左边」的变数. 注意, 你只能於左边使用一个变数, 也不能指定给常数或复杂的运算式. |

---

### + 加法运算子 (addition operator):

| | |
|---|---|
| | |

| | |
|---|---|
| 范例 | x + 7 |
| 值 | 左边值加上右边值的总和 |
| 说明 | 把右边运算式的值加上左边运算式的值. 对整数 (int) 型态值来说, 就表示数值总和. 对字串 (string) 来说, 表示右边的值接在左边的值後面 ("a"+"b" 的值是 "ab"). 这个运算子不改变任何原始值 (即变数 x 维持原来的值). |

---

- 减法运算子 (subtraction operator):

| | |
|---|---|
| 范例 | x - 7 |
| 值 | 左边运算式的值减去右边的 |
| 解释 | 除了它是减法以外, 与加法的特性相同. |
| 字串 | "ab" - "b" 的值是 "a". |

---

* 乘法运算子 (multiplication operator):

| | |
|---|---|
| 范例 | x*7 |
| 值与说明 | 除了这个作数学乘法之外, 与加法、减法相同. |

---

/ 除法运算子 (division operator):

| | |
|---|---|
| 范例 | x/7 |
| 值与说明 | 同上 |

---

+= 加法指定运算子(additive assignment operator):

| | |
|---|---|
| | |

| | |
|---|---|
| 范例 | x += 5 |
| 值 | 与 x + 5 相同 |
| 说明 | 它把左边的变数值和右边的运算式值加起来, 把总和指定给左边的变数. |
| 例如 | 如果 x = 2... x += 5 指定 7 值给变数 x. 整个运算式的值是 7. |

---

-= 减法指定运算子 (subtraction assignment operator):

| | |
|---|---|
| 范例 | x-=7 |
| 值 | 左边的值减去右边的值. |
| 说明 | 除了减法以外, 与 += 相同. |

---

*= 乘法指定运算子 (multiplicative assignment operator):

| | |
|---|---|
| 范例 | x *= 7 |
| 值 | 左边的值乘上右边的. |
| 说明 | 除了乘法以外, 与 -= 和 += 相似. |

---

/= 除法指定运算子 (division assignment operator):

| | |
|---|---|
| 范例 | x /= 7 |
| 值 | 左边变数的值除以右边的值. |
| 说明 | 除了除法以外, 同上. |

---

++ 後 / 前增加运算子 (post/pre-increment operators):

| | |
|---|---|
| 范例 | i++ 或 ++i |
| 值 | i++ 的值是 i<br>++i 的值是 i+1 |
| 说明 | ++ 改变 i 的值, 将 i 加上 1. 但是, 运算式本身的值是多少, 要看你把 ++ 摆在哪里. ++i 是前增加运算子. 这表示它的增加在给予值「之前」. i++ 是後增加运算子. 它计算在 i 增加之前. 重点在哪？好, 目前这对你来说无关紧要, 但是你应该记住它代表的意思. |

---

-- 後 / 前减少运算子 (post/pre-decrement operators):

| | |
|---|---|
| 范例 | i-- 或 --i |
| 值 | i-- 的值是 i<br>--i 的值是 i 减掉 1 |
| 说明 | 除了是减法以外, 就像 ++ |

---

== 相等运算子 (equality operator):

| | |
|---|---|
| 范例 | x == 5 |
| 值 | 真或伪 (非 0 或 0) |
| 说明 | 它不更改任何值, 但是<br>如果两个值相等就传回真.<br>如果两边不相等则传回伪. |

---

!= 不等运算子 (inequality operator):

| | |
|---|---|
| 范例 | x != 5 |

| | |
|---|---|
| 值 | 真或伪 |
| 说明 | 如果左边的运算式不等於右边的运算式就传回真. 如果它们相等则传回伪. |

---

> 大於运算子 (greater than operator):

| | |
|---|---|
| 范例 | x > 5 |
| 值 | 真或伪 |
| 说明 | 只有在 x 大於 5 时为真<br>如果相等或小於就为伪 |

---

< 小於运算子 (less than operator)
>= 大於或等於运算子 (greater than or equal to operator)
<= 小於或等於运算子 (less than or equal to operator):

| | |
|---|---|
| 范例 | x < y  x >= y  x <= y |
| 值 | 真或伪 |
| 说明 | 与 > 相似, 除了<br>< 如果左边小於右边就为真<br>>= 如果左边大於「或等於」右边则为真<br><= 如果左边小於「或等於」右边就为真 |

---

&& 逻辑与运算子 (logical and operator)
|| 逻辑或运算子 (logical or operator):

| | |
|---|---|
| 范例 | x && y  x \|\| y |
| 值 | 真或伪 |
| 说 | 如果右边的值和左边的值是非零值, && 为真.<br>如果任何一边是伪, 则 && 为伪. |

| | |
|---|---|
| 明 | 对 || 来说, 只要两边任何一个值是真, 则为真. 只有两边都是伪值时, 才为伪. |

---

! 否定运算子 (negation operator)

| | |
|---|---|
| 范例 | ! x |
| 值 | 真或伪 |
| 说明 | 如果 x 为真, 则 !x 为伪<br>如果 x 为伪, !x 就为真. |

---

底下有两个更复杂的运算子, 在此为了存在而存在. 如果它们让你一头雾水也别挂心.

---

-> 呼叫运算子 (the call other operator)

| | |
|---|---|
| 范例 | this_player()->query_name() |
| 值 | 被呼叫函式的传回值 |
| 说明 | 它呼叫右边这个函式, 而这个函式位於运算子左边的物件之内. 左边的运算式「必须」是一个物件, 而右边的运算式「必须」是函式的名字. 如果物件之中没有这个函式, 它会传回 0 (更精确一点, 没有定义 (undefined) ). |

---

? : 条件运算子 (conditional operator)

| | |
|---|---|
| 范例 | x ? y : z |
| 值 | 上面的例子里, 如果 x 为真, 其值为 y<br>如果 x 为伪, 其值为运算式 z |
| | |

| 说明 | 如果最左边的值为真, 这整个运算式的值就是中间的运算式. 不然, 就把整个运算式的值定为最右边的运算式. |
| --- | --- |

---

相等 (equality) 的注解:

大家所犯的一种很难除错、很糟糕的错误是把该写 == 的地方写成 =. 因为运算子有它的传回值, 这两种情况都能进行计算. 换句话讲, 这情形不会产生错误讯息. 但是这两者的值大不相同. 例如:

```
if(x == 5)    if(x = 5)
```

如果 x 是 5, 则其值为真. 反之则否.
x = 5 的值为 5 (所以它永远为真).
if 叙述会判断 () 之中的运算式是真还是伪, 所以如果你把 = 错当成 == , 你就会得到永远为真的运算式. 你会扯掉许多根头发, 也搞不清楚到底是为什麼出错 :)

---

译者: Spock of Final Frontier 98.Jan.26.

---

回上一页

---

# 基础 LPC

作者: Descartes of Borg
第一版: 23 april 1993
第二版: 10 july 1993

## 第七章: 流程控制 (flow control)

### 7.1 回顾变数

藉由 =、+=、-=、++、-- 等运算式, 可以指定或更改变数的值. 这些运算式可以与 -、+、*、/、% 结合使用. 但是, 到目前为止, 我们只告诉你如何用函式, 以线性的方式写出这些. 例如:

```
int hello(int x) {
    x--;
    write("嗨, x 是 "+x+".\n");
    return x;
}
```

你应该知道怎麼写出这个函式并了解它. 不过, 如果你只想於 x = 1 时显示 x 的值怎麼办？ 不然, 如果你想在传回 x 之前, 一直显示出 x 的值直到 x = 1 又要怎麼做？ LPC 使用的流程控制与 C 和 C++ 并无二致.

### 7.2 LPC 流程控制叙述

```
if(运算式) 指令;

if(运算式) 指令;
else 指令;

if(运算式) 指令;
else if(运算式) 指令;
else 指令
```

```
while(运算式) 指令;

do { 指令; } while(运算式);

switch(运算式) {
   case (运算式): 指令; break;
   default: 指令;
}
```

我们讨论这些东西之前, 先谈一下什麽是运算式和指令. 运算式是任何有值的东西, 像是变数、比较式 (像 x > 5, 如果 x 是 6 或 6 以上, 则其值为 1, 不然其值为 0)、指定式 (像 x += 2). 而指令是任何一行单独的 LPC 码, 像是函式呼叫、值指定式 (value assignment)、值修改式 (value modification) .....等等.

你也应该知道 && 、||、==、!=、！这些运算子. 它们是逻辑运算子. 当条件为真时, 它们传回非零值, 为伪时则传回 0. 底下是运算式值的列表:

```
(1 && 1)  值: 1 (1  和 1)
(1 && 0)  值: 0 (1  和 0)
(1 || 0)  值: 1 (1  或 0)
(1 == 1)  值: 1 (1  等於 1)
(1 != 1)  值: 0 (1  不等於 1)
(!1)      值: 0 ( 非 1)
(!0)      值: 1 ( 非 0)
```

使用 && 的运算式中, 如果要比较的第一项测试值为 0, 则第二项永远不会测试之. 使用 || 时, 如果第一项为真 (1), 就不会测试第二项.

---

## 7.3 if()

---

我们介绍第一个改变流程控制的运算式是 if(). 仔细看看底下的例子:

```
1 void reset() {
2     int x;
3
4     ::reset();
```

```
5     x = random(100);
6     if(x > 50) set_search_func("floorboards", "search_floor");
7 }
```

- 每一行的编号仅供参考.
- 在第二行, 我们宣告一个称为 x 的整数型态变数. 第叁行则优雅地留下一行空白, 以明示宣告结束和函式码开始的界线. 变数 x 只能在 reset() 函式中使用.
- 第四行呼叫 room.c 中的 reset().
- 第五行使用 driver 外部函式的 random() 以传回一个随机数字, 此数字介於 0 到参数减一. 所以在此我们想得到一个介於 0 到 99 的数字.
- 第六行中, 我们测试运算式 (x>50) 的值, 看它是真是伪. 如果为真, 则呼叫 room.c 的函式 set_search_func(). 如果为伪, 就不可能执行呼叫 set_search_func() .
- 第七行, 函式将 driver 的控制权交回呼叫此函式的函式 (在这个例子中, 呼叫 reset() 的是 driver 自己) , 也没有传回任何值.

如果你想执行一个以上的指令, 你必须按照以下的方法来做:

```
if(x>50) {
   set_search_func("floorboards", "search_floor");
   if(!present("beggar", this_object())) make_beggar();
}
```

注意运算式为真时, 要执行的指令以 {} 包围起来. 这个例子里, 我们再次呼叫 room.c 中的 set_search_func() 来设定一个函式 (search_floor()) , 这个函式稍後被你设定为: 玩家输入 "search floorboards" 时, 呼叫 search_floor(). (注: 这种例子要看 mudlib 而定. Nightmare 有这个函式呼叫, 其他 mudlib 可能会有类似的东西, 也可能完全没有这一类用途的函式) 接着, 另一个 if() 运算式检查 (!present("beggar", this_object())) 运算式是否为真. 测试运算式中的 ! 改变它後面运算式的真伪. 在此, 它改变外部函式 present() 的真伪值. 在此, 如果房间里有个乞丐, present() 就传回乞丐这个物件 (this_object()), 如果没有乞丐, 则传回 0. 所以, 如果房间里面还有个活乞丐, (present("beggar", this_object())) 的值就会等於乞丐物件 (物件资料型态) , 不然它会传回 0. ! 会把 0 变成 1 , 把任何非零值 (像是乞丐物件) 变成 0. 所以, 房间里没有乞丐时, 运算式 (!present("beggar", this_object())) 为真, 反之, 有乞丐为 0. 如果房间里没乞丐, 它呼叫你房间码中定义的函式来制造一个新的乞丐, 并放进房间.

(如果房间中已经有一个乞丐, 我们不想多加一个 :) )

当然, if() 常常和一些条件一起出现 :). LPC 里, if() 叙述的正式写法为:

```
if(运算式) { 一堆指令 }
else if(运算式) { 一堆指令 }
else { 一堆指令 }
```

这样表示:

如果运算式为真, 执行这些指令.
不然, 如果第二个运算式为真, 执行第二堆指令.
如果以上皆伪, 执行最後一堆指令.

你可以只用 if() :

```
if(x>5) write("Foo,\n");
```

跟着一个 else if():

```
if(x > 5) write("X 大於 5.\n");
else if(x >2) write("X 小於 6, 大於 2.\n");
```

跟着 else:

```
if(x>5) write("X 大於 5.\n");
else write("X 小於 6.\n");
```

或是把上面列出来的东西全写出来. 你有几个 else if() 都没关系, 但是你必须有一个 if() (也只能有一个), 也不能有一个以上的 else . 当然, 上面那个乞丐的例子中, 你可以在 if() 叙述中重复使用 if() 指令. 举例来说,

```
if(x>5) {
   if(x==7) write("幸运数字 !\n");
   else write("再试一次.\n");
}
else write("你输了.\n");
```

# 7.4 叙述: while() 和 do {} while()

原型:

```
while(运算式) { 一堆指令 }
do { 一堆指令 } while(运算式);
```

这两者让你在运算式为真时, 一直重复执行一套指令. 假设你想设定一个变数等於玩家的等级, 并持续减去随机的金钱数量或可承受伤害值 (hp, hit points) 直到该等级变数为 0 (这样一来, 高等级的玩家失去的较多). 你可能会这样做:

```
1  int x;
2
3  x = (int)this_player()->query_level();  /* 这行内容等一下会解释
*/
4  while(x > 0) {
5      if(random(2)) this_player()->add_money("silver", -random(50));
6      else this_player()->add_hp(-(random(10));
7      x--;
8  }
```

第叁行中呼叫的 this_player()->query_level() 运算式 (译注: 之後内容遗失, 在此由译者补充) 的意义: 呼叫 this_player() 外部函式, this_player() 传回一个物件, 为正在呼叫此函式的玩家物件. 再呼叫此玩家物件中的 query_level() 函式. (译注: 补充结束)

在第四行, 我们开始一个回圈, 只要 x 大於 0 就重复执行.
我们可以用另一种写法:

```
while(x) {
```

(译注: 以下遗失, 由译者补充)
由於 x 本身稍後会一直减 1 直到到 x = 0 , 所以 x = 0 时也是伪值 (为 0).
第五行以 random(2) 随机传回 0 或 1. 如果它传回 1 (为真), (译注: 补充完毕) 则呼叫玩家物件的 add_money() 将玩家身上的银币随机减少 0

到 49 枚.
在第六行, 如果 random(2) 传回 0, 我们呼叫玩家物件中的 add_hp() 函式来减少 0 到 9 点的可承受伤害.
第七行里, 我们把 x 减 1.
第八行执行到 while() 指令的终点, 就回到第四行看 x 是否还大於 0 . 此回圈会一直持续执行到 x 小於 1 才结束.

但是, 你也许想在你执行一些指令「之後」再测试一个运算式. 比如用上面的例子, 如果你想让每个人至少执行到一次指令, 甚至还不到测试的等级:

```
int x;
x = (int)this_player()->query_level();
do {
   if(random(2)) this_player()->add_money("silver", -random(50));
   else this_player()->add_hp(-random(10));
   x--;
} while(x > 0);
```

这个例子真的很奇怪, 因为没几个 mud 会有等级为 0 的玩家. 而且, 你可以修改前面例子中的测试条件做到同样的事. 不管如何, 这个例子只是要展现出do {} while() 的如何工作. 如你所见, 此处在回圈开始的时候没有初始条件 (在此不管 x 的值为何, 立刻执行) , 回圈执行完之後才测试. 这样能保证回圈中的指令至少会执行一次, 无论 x 为何.

---

## 7.5 for() 回圈

---

原型:

```
for(初始值 ; 测试运算式 ; 指令) { 指令 }
```

初始值:

让你设定一些变数开始的值, 用於回圈之内. 此处可有可无.

测试运算式:

与 if() 和 while() 的运算式相同. 当这一个 (或一些) 运算式为真时, 执行回圈. 你一定要有测试运算式.

指令:

一个 (或一些) 运算式, 於每个回圈执行完毕之後执行一次. 此处可有可无.

注:

for(;运算式;) {}

与

while(expression) {}

「 完 全 相 同 」

范例:

```
1   int x;
2
3   for(x= (int)this_player()->query_level(); x>0; x--) {
4       if(random(2)) this_player()->add_money("silver", -random(50));
5       else this_player()->add_hp(-random(10));
6   }
```

这个 for() 回圈与前面 while() 的例子「完全相同」. 还有, 如果你想初始化两个变数:

for(x=0, y=random(20); x<y; x++) { write(x+"\n"); }

在此, 我们初始化 x 和 y 两个变数, 我们把它们用逗号分开来. 你可以在 for() 叁个部分的运算式中如此使用.

---

## 7.6 叙述: switch()

---

原型:

```
switch(运算式) {
    case 常数: 一些指令
    case 常数: 一些指令
```

```
    ......
    case 常数: 一些指令
    default: 一些指令
}
```

这样有点像 if() 运算式, 而且对 CPU 也好得多, 但是 switch() 很少有人使用它, 因为它看起来实在很复杂. 但是它并非如此.

第一点, 运算式不是测试条件. case 才是测试. 用普通的话来读:

```
1   int x;
2
3   x = random(5);
4   switch(x) {
5       case 1: write("X is 1.\n");
6       case 2: x++;
7       default: x--;
8   }
9   write(x+"\n");
```

就是:

设定变数 x 为一个 0 到 4 的随机数字.
x = 1 的 case 中, 显示 x 的值, 将 x 加上 1 之後再将 x 减 1.
x = 2 的 case 中, 将 x 加上 1 之後再减 1.
其他情形下, x 减 1.
显示 x 的值.

switch(x) 基本上告诉 driver, 变数 x 的值是我们想配合各个 case 的情形. 当 driver 找到一个能配合的 case 时, 这个 case 「以及所有在它之後」的 case 都会执行. 你可以使用 break 指令, 在执行一个 case 之後跳出 switch 叙述, 就像其他流程控制叙述一样. 稍後会解释这一点. 只要 switch() 流程还没中断, 任何 x 值都会执行 default 叙述. 你可以在 switch 叙述中使用任何资料型态:

```
string name;

name = (string)this_player()->query_name();
switch(name) {
```

```
        case "descartes": write("You borg.\n");
        case "flamme":
        case "forlock":
        case "shadowwolf": write("You are a Nightmare head arch.\n");
        default: write("You exist.\n");
    }
```

对我来说, 我会看到:

```
You borg.
You are a Nightmare head arch.
You exist.
```

Flamme、Forlock 、或 Shadowwolf 会看到:

```
You are a Nightmare head arch.
You exist.
```

其他人会看到:

```
You exist.
```

---

## 7.7 改变函式的流程和流程控制叙述

---

以下的指令:

return　continue　break

能改变前面提过的那些东西, 它们原本的流程.
首先,

return

一个函式中, 不管它出现在哪里, 都会终止执行这个函式并将控制权交回呼叫这个函式的函式. 如果这个函式「不是」无传回值 (void) 的型态, 就必须在 return 叙述之後跟着一个传回值. 一个绝对值函式长得大概像这样:

```
int absolute_value(int x) {
```

```
    if(x>-1) return x;
    else return -x;
}
```

第二行里, 函式终止执行, 并回到呼叫它的函式. 因为在此, x 已经是正整数.

continue 在 for() 和 while() 叙述中用得最多. 它停止目前执行的回圈, 把回圈送回开头执行. 例如, 你想要避免除以 0 的情况:

```
x= 4;
while( x > -5) {
    x--
    if(!x) continue;
    write((100/x)+"\n");
}
write("完毕.\n")
```

你会看到以下的输出:

```
33
50
100
-100
-50
-33
-25
完毕.
```

为了避免错误, 每一次回圈都检查 x, 确定 x 不为 0. 如果 x 是 0, 则回圈回到开头处的测试运算式, 并不终止目前的回圈.

用 for() 运算式来说就是:

```
for(x=3; x>-5; x--) {
  if(!x) continue;
  write((100/x)+"\n");
}
write("完毕.\n");
```

这样执行起来差不了多少. 注意, 这样子跟前面输出的结果一模一样. 当 x = 1, 它测试 x 是否为 0, 如果不是, 就显示 100/x, 然後回到第一行, 将 x 减 1, 再检查 x 是否是 0 , 如果为 0, 回到第一行并把 x 再减 1.

break

它停止执行流程控制叙述. 不管它出现在叙述里面的任何地方, 程式控制会结束回圈. 所以, 如果在上面的例子中, 我们把 continue 换成 break, 则输出的结果会变成像这样:

```
33
50
100
完毕.
```

continue 最常用於 for() 和 while() 叙述. 但是 break 常用於 switch().

```
switch(name) {
    case "descartes": write("You are borg.\n"); break;
    case "flamme": write("You are flamme.\n"); break;
    case "forlock": write("You are forlock.\n"); break;
    case "shadowwolf": write("You are shadowwolf.\n"); break;
    default: write("You will be assimilated.\n");
}
```

下面这个函式跟上面的一样:

```
if(name == "descartes") write("You are borg.\n");
else if(name == "flamme") write("You are flamme.\n");
else if(name == "forlock") write("You are forlock.\n");
else if(name == "shadowwolf") write("You are shadowwolf.\n");
else write("You will be assimilated.\n");
```

但是 switch 叙述对 CPU 比较好.
如果这些指令放在多层巢状 (nested) 的叙述中, 它们会改变最近的叙述.

---

## 7.8 本章总结

---

这一章讲的东西实在是太多了, 但是它们马上就用得到. 你现在应该完

全了解 if()、for()、while()、do{} while()、switch(), 也该完全了解如何使用 return、continue、break 改变它们的流程. 使用 switch() 要比一大堆 if() else if() 来得有效率, 所以应该尽量使用 switch() . 我们也向你介绍过怎麽呼叫其他物件中的函式. 不过, 以後会详细解释这个主题. 你现在应该能轻轻松松写出一个简单的房间 (如果你已经读过你 mudlib 有关建造房间的文件)、简单的怪物、其他简单的物件.

---

译者: Spock of Final Frontier 98.Feb.1.

---

回上一页

---

# 基础 LPC

作者: Descartes of Borg
第一版: 23 april 1993
第二版: 12 july 1993

## 第八章: 「物件」资料型态

### 8.1 回顾

你现在应该能从你自己的物件中呼叫函式. 你也应该清楚, 至少在一开始物件载入记忆体的时候, 你物件中的 create() (或 reset() ) 函式会被呼叫, 而你的 reset() 函式会一直被重复呼叫, 让你可以写些程式码来更新你的房间状况. 注意一下, 你的物件中不一定要有这两个函式. driver 会先检查你的物件中有没有这些函式. 如果没有, 也不会怎麼样. 你也已经认识 void (无传回值), int (整数), string (字串) 这叁种资料型态.

### 8.2 物件是一种资料型态

在这一章里面, 你将会认识一种更复杂的资料型态——物件. 一个物件变数指向一个已经载入 driver 记忆体的真正物件. 宣告物件变数的方法跟宣告其他资料型态的变数一样:

```
object ob;
```

不过它不同的地方在於你不能在它身上用 +、- 、+=、-=、* 、/ (把一只怪物除以另一只怪物到底有啥意义 ? ). 而且, 像是 say() 和 write() 外部函式只要字串或整数, 你就不能 write() 或 say() 它们 (再次声明, 说一只怪物是啥意思 ? ). 但是你可以将它们用於其他 LPC 重要的外部函式上.

### 8.3 外部函式: this_object()

这个外部函式传回一个物件, 是正在执行 this_object() 的物件. 换句话说, 在一个档案里, this_object() 就是你的档案物件复制出去的拷贝或是

继承这个档案的其他档案. 当你正在撰写一个会被别的档案继承的档案, this_object() 就很有用. 假设你正在写你自己的 living.c , user.c 和 monster.c 会继承它, 但是 living.c 不可能会独自使用, 它只用来被这两个物件继承. 你想要把设定玩家等级的 set_level() 函式记录下来, (但是你不想记怪物的).
你可能会这样做:

```
void set_level(int x) {
   if(this_object()->is_player()) log_file("levels", "foo\n");
   level = x;
}
```

既然 living.c 或 living.c 继承的档案都没有定义 is_player(), 我们就假设 if(is_player()) 会导致一个错误, 因为 driver 在你的档案里、你继承的档案中都找不到 is_player() 函式. 因为你的档案是被别的档案继承之故, this_object() 让你能使用最後成品中可能拥有 (或没有) 的函式而不会出现错误.

---

## 8.4 呼叫其他物件中的函式

---

这当然是向你介绍物件资料型态最重要的特色. 它让我们能使用其他物件中的函式. 前面的范例里, 你已经能找出一个玩家的等级、减少他们身上的钱、他们有多少可承受伤害点数.
有两种方法可以呼叫其他物件中的函式:

```
物件->函式(参数)
call_other(物件, "函式", 参数);
```

范例:

```
this_player()->add_money("silver", -5);
call_other(this_player(), "add_money", "silver", -5);
```

某些情形下 (很概略的说法) , 游戏只是由玩家命令触发的一连串函式呼叫. 当一个玩家开始一串函式呼叫时, 这个玩家就是 this_player() 外部函式所传回的物件. 所以, 因为 this_player() 可以由触发事件的人决定, 你要小心你用 this_player() 呼叫函式的地方在哪里. 你通常会把它摆在最後一个重要的区域函式—— init() 里 (我们已经提过 create() 和

reset() ).

## 8.5 区域函式: init()

任何时候, 一个活着的东西碰到一个物件 (进入一个新的房间, 或其他物件进入同一个房间) , 就会呼叫此物件新遇到所有物件里面的 init() 函式. 在此, 你可以加上一些玩家可以使用的命令. 以下是一朵花的 init() 函式范例.

```
void init() {
   ::init();
   add_action("smell_flower", "smell");
}
```

上面呼叫 smell_flower() 函式. 所以你应该有个 smell_flower() 函式长得像这样:

```
1 int smell_flower(string str);       /* 玩家动作的函式是整数型态 */
2
3 int smell_flower(string str) {
4    if(str != "flower") return 0;     /* 玩家闻的不是这朵花 */
5    write("你闻了这朵花.\n");
6    say((string)this_player()->query_cap_name()+"闻了闻花.\n");
7    this_player()->add_hp(random(5));
8    return 1;
9 }
```

- 第一行, 我们宣告函式.
- 第叁行, 开始 smell_flower(). str 是跟在玩家命令之後的任何东西 (不包括第一个空白字元).
- 第四行, 检查玩家输入的是否为 "smell flower". 如果玩家输入的是 "smell cheese", 则 str 就是 "cheese". 如果闻的不是花, 就传回 0, 让 driver 知道不该呼叫这个函式. 如果玩家身上有块乳酪, 乳酪也有个 smell 指令的话, driver 之後会呼叫乳酪的函式. driver 会持续呼叫同样是 smell 的命令, 直到有一个传回 1 为止. 如果它们都传回 0, 则玩家就看到「什麽？」
- 第五行, 呼叫 write() 外部函式. write() 把传入给它的字串印出来给 this_player() . 所以, 只要输入 "smell flower" 的玩家都会看到「你

闻了这朵花.」

- 第六行, 呼叫 say() 外部函式. say() 印出闻花动作的字串, 我们需要呼叫 this_player() 的 query_cap_name() 函式. 这样子碰上隐形的玩家会印出「某个人」(或像是隐形的东西), 而且会把第一个字元转为大写 (capitalize).
- 第七行, 我们呼叫 this_player() 物件中的 add_hp() 函式, 因为我们想在闻了花之後对玩家作一点治疗 (注: 别把这些程式码写在你的 mud 里, 管理 mud 平衡的人会毙了你).
- 第八行, 我们把游戏的控制交回给 driver, 传回 1 让 driver 知道它呼叫的函式正确.

---

## 8.6 在你的房间加上物件

---

现在, 使用物件资料型态, 你可以把怪物加进房间里面:

```
void create() {
    ::create();
    set_property("light", 3);
    set("short", "Krasna 广场");
    set("long", "欢迎来到 Praxis 镇的中央广场.\n");
    set_exits( ({ "d/standard/hall" }), ({ "east" }) );
}

void reset() {
    object ob;

    ::reset();
    if(present("guard")) return;     /* 如果已经有一位警卫, */
    ob = new("/std/monster");        /* 就别再增加一位 */
    ob->set_name("guard");
    ob->set("id", ({ "guard", "town guard" }) );
    ob->set("short", "镇警卫");
    ob->set("long", "它看守着 Praxis.\n");
    ob->set_gender("male");
    ob->set_race("human");
    ob->set_level(10);
    ob->set_alignment(200);
    ob->set_humanoid();
```

```
    ob->set_hp(150);
    ob->set_wielding_limbs( ({ "right hand", "left hand" }) );
    ob->move(this_object());
}
```

现在, 大多数的 mud 在此都大不相同. 前面提过, 有的 mud 把这些东西写在一个独立设定的怪物物件里. 原始模式的 mud 最後要呼叫怪物物件中的 move() 来把它搬进房间 (this_object() ) 里. 在精简模式的 mud 里, 你呼叫需要两个参数的 move_object() 外部函式, 这两个参数是: 要搬动的物件和要放东西进去的物件.

---

## 8.7 本章总结

---

行文至此, 你现在应该有相当的知识来撰写一些很棒的东西. 当然, 我一直强调你真的需要去阅读如何在你 mud 写程式的说明文件, 它们会详细说明在什麼种类的物件里拥有哪些函式可以呼叫. 无论你对 mudlib 的知识有多少, 你已经有足够的知识了解如何给玩家一些额外的事情做, 像是闻闻花、贴东西之类的事. 现在你应该能忙於撰写程式. 但是此刻, 事情看起来变得枯燥沉闷, 这表示你该进入下一阶段、更深入的时间到了. 现在让你自己撰写一个小区域. 尽量使用你 mud room.c 里头所有的特殊函式 (找找别人觉得用都用不到的冷僻文件) . 加上一堆简洁的动作. 创造一些含有魔力的武器, 其魔力会渐渐消失. 以上这些你现在应该都能写得出来. 一旦这些东西对你来说都变成例行公事, 就是你开始学习中阶课程的时候. 注意, 只有很少人能真正进入中阶课程. 如果你全部都做完, 我告诉你, 你在 mud 中能做到的领域只在少数. 这不仅是因为其他许多领域很困难, 也因为有一些已经超越此领域的人充满了傲慢, 而且极少传播这些知识. 秘诀在於: 强迫你自己, 并想一些你觉得不可能做到的事. 如果你问某个人怎麼做 X, 而他们跟你说那个不可能做到, 就自己想办法利用实验把它写出来.

George Reese
Descartes of Borg
12 july 1993
borg@hebron.connected.com
(译按: 已改为 borg@imaginary.com )
Descartes@Nightmare (intermud)
Descartes@Igor (not intermud)

---

译者: Spock of Final Frontier  98.Feb.2.

回上一页

# 中阶 LPC

Descartes of Borg
November 1993

## 目录





翻译: Spock of the Final Frontier 98.Mar.19.

回到上一页

# 中阶 LPC

Descartes of Borg
November 1993

## 第一章: 简介

### 1.1 基础 LPC

阅读此课本的人应该读过基础 LPC 课本或是够熟悉 mud 世界的程式写作. 不只是能建造房间和撰写区域内有关的物件而已, 也该清楚自己写出来的程式在执行的时候到底在做什麽. 如果你觉得你的程度还不到, 就回去看完基础 LPC 再来看中阶 LPC 课本. 如果你达到要求, 你会发现你在此读到的东西对你比较有意义.

### 1.2 中阶 LPC 课本的目标

此份介绍性的课本是为了对 LPC 一无所知的人, 让他们有能力在任何 LPMud 写出好的游戏世界. 对 LPC 和建设 LPMud 来说, 自然要比建造房间、护甲、怪物、武器来得艰深. 当你进入更复杂的概念, 例如公会; 或想更深入你的游戏世界, 你会发现基础 LPC 详细说明的概念里面没有这些东西. 中阶 LPC 的设计是把你从简单的世界建造过程, 带到完全了解 LPC 作为 LPMud 世界建造者的知识. 撰写 mudlib 本身的工作留到後面再讲. 读完这份课本, 并实际撰写一些实验性的程式码之後, 你们读者应该能写出合乎任何设计或想法的游戏物件, 只要这样我的目的就达到了.

### 1.3 概观

到底多了什麽东西？ 呃, 你们大部份都知道 LPC 支援映射 (mapping) 和阵列 (array), 也曾问过我基础 LPC 为什麽没有详细说明它们. 我觉得那些概念超出我在基础 LPC 课本里面尝试讲述的范围, 而比较适合放在这份课本里. 不过, 新工具都棒极了, 而重要的是, 你可以用新工具做啥. 基础 LPC 课本的目标是让你能建造够格的 LPMud 区域, 不需要映射 (mapping) 和阵列 (array) 就可以办到. 这份课本的目标是让你能在你

的区域中实现任何点子, 这样就需要映射和阵列的知识.

LPMud 中, 任何你想写的点子都可能实现. 令人惊讶的是, LPC 语言非常适合实现这些目标. 你无法写出你的点子的原因是: 你对 LPC 的知识、不合的 mudlib、你的 Mud 主题、mud 管理的政策. 这份课本没办法把你工作的 mudlib 变得更好, 也无法改变你的 Mud 主题或 mud 的管理政策. 千万别认为 LPC 没办法做到你想做的事. 如果你的点子不容於 mud 主题或是管理政策, 那就是你的点子无法在你现在的 mud 实现. 如果 mudlib 不合, 告诉负责 mudlib 的人, 在 mudlib 层面上可以做哪些事来增进它. 你会很惊讶, mudlib 中实际上有很多你不知道的东西. 更重要的是, 读过此课本後, 你应该能读懂你 mud 的 mudlib 程式码, 了解每一行程式在做啥. 你目前应该还没有能力自己写出这种程式码, 但是至少你可以了解 mudlib 层次在做什麽.

这份课本开始时, 讨论 LPMud driver 到底在做些什麽. 这份课本的一个好处是, 里面所讲的东西跟 driver 和 mudlib 无关 (除了 Dworkin Game Driver). 对於游戏 driver 的章节不会深入实际的 driver 程式架构, 而是所有游戏 driver 跑 mud 基本上所要做的事.

接着, 我讨论每个人都想多了解一些的题目 ---- 阵列 (array) 和映射 (mapping). 要了解映射这种资料型态, 大概是最简单也是最困难的事. 大致上, 映射算是一种复杂的阵列, 要讨论映射之前, 你应该先了解阵列. 一旦你了解它们, 映射实际上要比阵列好用得多. 无论快慢, 用你最多的的时间来学习这一章, 因为本章大概是这份课本里头最困难, 也是最有用的一章.

之後, 有一小章讨论 LPC pre-compiler, 在你的程式码送给 compiler 之前, 你可以用它来编排你的程式码. 虽然我在这里介绍得很可怕, 这章恐怕是这份课本中最简单的一章. 所以我把它摆在介绍映射和阵列的那一章之後.

接着重新介绍字串 (string) , 更详细地教你如何处理字串, 像是分断字串这种高级的命令处理技巧. 只要你很了解阵列, 这章应该很简单.

再下一章是这本课本第二重要的部分. 如果你想超越中阶的水准、深入 mudlib 程式码, 这可能是最重要的部分. 这章深入介绍 LPC 继承 (inheritance) 的复杂观念. 既然这份课本的目标并不是教你设计 mudlib 的程式, 这章不会详细讨论物件导向程式设计. 了解本章, 会让你体会一些物件导向程式设计的内涵, 也让你能够僭越 (overriding) 其他函式以

写出更复杂的物件, 并定义出你自己的角色基础职业.

最後, 课本以简略地讨论程式码除错作结. 这不是很重要的一章, 但是这样也表示此章不只是补充你目前所学的知识而已.

## 1.4 此课本没有的东西

对某些人来说, 此份课本最大的、 也是政策性的遗漏就是「投影」(shadow). 我从来没有看过使用投影是最好或最有效率的例子. 不过, 这样也不代表投影一无是处. 我在这份课本里不介绍投影的理由是, 学习 LPC 的人, 最好在碰上投影以前, 先从此课本学得一些观念, 并花上时间去熟悉这些观念. 这样一来, 我觉得学习 LPC 的人会有能力决定是否要使用投影. 我会在以後的课本里讨论投影.

如果你经常使用很多的投影, 请别认为上面这段文字是针对你的批评. 我也曾经看过投影有很多优秀的用途. 不过, 投影并不是一个完成工作的好方法, 所以投影并不适合这份中阶课本的目标.

我也删掉了讨论系统安全和物件导向程式设计的部份. 这两者很明显都是讨论 mudlib 方面的题目. 不过很多人大概会反对我不讨论物件导向程式设计的作法. 我决定把这个课题留到以後再说, 因为大多数区域设计者是为了创作而撰写程式码, 而不是为了资讯理论. 在中阶和基础的课本里, 我决定只在实际 LPC 程式设计上可以直接应用的地方讨论物件导向程式设计的理论. 对於想撰写一个庞大 mudlib 的 LPC 老手而言, 理论可能要实用得多. 不过以这份课本的目标来说, 讨论物件导向程式设计只是个让人打瞌睡的题目. 我计划在下一份课本里多讨论这个理论.

## 1.5 总结

LPC 不难学习. 虽然比不上其他大部分电脑语言所常做的工作, 令人惊异的是它非常强大, 在建造 MUD 这种游戏的工作上, 也没其他语言比得上它. 对初学者来说, 它让你易於学习, 甚至在你还不知道在做什麼的时候, 就能写出有用的物件. 对中阶的人来说, 它让你的任何点子变成文字化的虚拟实境. 对高阶的人来说, 它的物件导向特点, 可以让你建造一个 internet 上最受喜爱的游戏. 你唯一所受到的限制, 是你了解多少东西. 而进一步学习并不需要资讯学位.

译者: Spock of the Final Frontier 98.Jul.19.

回到上一页

# 中阶 LPC

Descartes of Borg
November 1993

## 第二章: LPMud driver

### 2.1 回顾基本的 driver/mudlib 间的互动

在基础 LPC 课本里, 你学到很多 mudlib 工作的方式, 尤其是关於你为了建造区域所撰写的物件. 而 mudlib 和 driver 间的互动讨论得并不多. 不过, 你应该知道 driver 做了以下的事:

- 1) 当一个物件第一次被载入记忆体, 原始模式 mud 的 driver 会呼叫 create(), 而精简模式 mud 会呼叫 reset(). 创作的人使 create() 或 reset() 给予物件初始值.
- 2) 每到游戏管理者设定的周期, driver 呼叫 reset() 函式. 这样让物件能重新产生怪物之类的东西. 请注意, 在精简模式的 mud 中, 同一个函式不但用於重新设定房间, 也用於设定初始值.
- 3) 任何时候, 一个活物件 (living object) 遇到另一个物件时, driver 呼叫新遇到物件的 init() 函式. 这样可以让新遇到的物件透过 add_action() 外部函式 (efun) 给予活物件可以执行的命令, 同样也可以执行其他的动作, 而这些动作是一个活物件碰到此一物件时所该发生的事.
- 4) driver 定义了一套称为外部函式的函式, 在游戏中所有的物件都可以使用它们. 举例来说, 常用的外部函式有: this_player(), this_object(), write(), say(), 以此类推.

### 2.2 driver 周期 (cycle)

driver 是执行游戏的 C 程式. 它的基本功能是接受外界的连线, 让人能登录 (login) 、 解译定义 LPC 物件和它们在游戏中作用的 LPC 程式码、接受使用者的输入并呼叫适当的 LPC 函式以配合事件发生. 它最简单的要素就是, 它是一个永不终止回圈 (loop).

一旦游戏启动, 并且正确地执行功能 (以後会在高阶 LPC 课本中讨论启动程序), driver 就进入一个回圈. 除非合法呼叫 shutdown() 外部函式, 或碰上臭虫让 driver 崩坏 (crash), 此回圈不会终止. 一开始, driver 控制任何新进的连线, 并把连线交给登录物件 (login). 之後, driver 把所有使用者输入的命令放入一个命令表 (table of commands), 此时已是 driver 的最後一个周期. 在组合命令表之後, 所有从 driver 最後一个周期排定要送给连线的讯息, 就送给使用者. 此时, driver 依序执行命令表中的命令, 并执行每个物件放在命令表中的各套命令. driver 在周期结束时, 呼叫每一个有 heart_beat() 函式的物件, 执行其中的 heart_beat() 函式. 最後, 执行所有等待的延迟呼叫 (call out). 本章不讨论连线控制, 本章焦点放在 driver 如何控制使用者命令、心跳 (heartbeat)、延迟呼叫.

## 2.3 使用者命令

如同 1.2 中所提, driver 在每个周期中, 把每一个使用者要执行的命令储存在命令表里. 命令表里头有执行此命令的活物件名称、给予活物件此一命令的物件、要执行此命令时所执行的函式. driver 把输入命令的物件当作是给予命令者. 大多数的时候, 这就是 this_player() 所传回的给予命令者.

driver 由有延迟命令的活物件表的头端开始, 接着执行命令, 呼叫这些活物件输入的命令相关的函式, 并传入给予命令者给函式的任何参数. 当 driver 由新的活物件所给的命令开始时, 给予命令者变数就改为新的活物件, 这样在命令开始依序执行函式时, this_player() 外部函式才能传回给予命令的物件.

来看看一个玩家的命令暂存区范例. 在一个叫做 Bozo 的玩家执行最後一个命令时, 他输入 "north" 和 "tell descartes 下次重新开机是什麼时候 ?". "north" 命令与 Bozo 所在房间里的 "Do_Move()" 函式相关 ("north" 命令由此房间的 set_exits() 外部函式自动设定). "tell" 命令并没有特别列在玩家所可以使用的命令中, 而在玩家物件中有一个叫做 "cmd_hook()" 的函式, 比对玩家可能输入的命令.

当 driver 处理到 Bozo, 给予命令者的变数就设定为 Bozo 这个物件. 然後, 看到 Bozo 输入 "north", 也看到与 "north" 相关的函式, 则 driver 呼叫 Bozo's_room->Do_Move(0) (Bozo 所在房间的 Do_Move() 函式). 因为 Bozo 只输入 "north" 命令, 没有加上参数, 所以用参数 0 传入此函

式. 此房间平常会呼叫一些它需要的函式, 此时 this_player() 外部函式所传回的物件就是 Bozo. 最後, 此房间物件会呼叫 Bozo 中的 move_player(), 之後呼叫 move_object() 外部函式. 这个外部函式负责改变一个物件的环境.

当一个物件的环境改变时, 会删除前一个环境中其他物件和前一个环境中对它加上的可用命令. 删除之後, driver 呼叫新环境和新环境中每一个物件的 init() 外部函式. 每一次呼叫 init() 时, Bozo 物件仍然是给予命令者. 所以此次移动所有的 add_action() 外部函式会加在 Bozo 身上. 完成所有的呼叫後, 控制权从 move_object() 交给 Bozo 的 move_player() 区域函式. move_player() 将控制权交回给旧房间的 Do_Move(), Do_Move() 传回 1 给 driver, 以表示此命令的动作完成. 如果 Do_move() 因为某些原因传回 0, 则 driver 会对 Bozo 显示 "什麽?" (或是你的 driver 所预设的错误命令讯息).

一旦第一个命令传回 1, driver 就继续处理 Bozo 的第二个命令, 过程就跟第一个一样. 请注意, driver 把 "tell descartes 什麽时候重新开机 ?" 的 "descartes 什麽时候重新开机 ?" 当作参数传给跟 tell 相关的函式. 这个函式决定要如何处理这个参数. 这个命令之後传回 1 或 0, driver 再继续处理下一个有延迟命令的活物件, 然後以同样的步骤处理全部有延迟命令的活物件, 执行它们的命令.

---

## 2.4 set_heart_beat() 和 call_out() 外部函式

---

一旦有延迟命令的物件其全部的命令执行完成後, driver 就继续呼叫所有 driver 列为有心跳之物件中的 heart_beat() 函式. 只要一个物件以非零参数呼叫 set_heart_beat() 外部函式 (视你的 driver 而定, 非零的数字也许很重要, 但是在大多的情况下为整数 1 ), set_heart_beat() 外部函式把呼叫 set_heart_beat() 的物件加在有心跳物件的列表上. 如果你以 0 为参数呼叫它, 它就把此物件从有心跳物件的表上删除.

心跳在 mudlib 里最常见的用途是治疗玩家和怪物、执行战斗. 一旦 driver 处理完命令列表, 它就开始看心跳列表, 呼叫表上每一个物件的 heart_beat(). 所以举例来说, 对玩家而言, driver 会呼叫玩家里面的 heart_beat() 以执行以下功能:

- 1) 让玩家变老
- 2) 依照治疗速率治疗玩家.

- 3) 检查四周是否有任何被人猎杀、正在猎杀人、或正在攻击人的物件
- 4) 如果第叁点成立, 开始攻击.
- 5) 其他需要每秒钟自动发生的事.

请注意, 有心跳的物件越多, mud 每个周期需要处理的时间也就越久. 有心跳的物件已知是 mud 贪求 CPU 时间最主要的因素.

call_out() 外部函式用於执行不需要像心跳一样常常发生、或只发生一次的计时函式呼叫. 延迟呼叫 (call out) 让你指定呼叫一个物件中的某个函式. 一般延迟呼叫的公式为:

```
call_out( func, time, args );
```

第叁个指定参数的参数并非必要. 第一个参数是一个字串, 代表被呼叫的函式名称. 第二个参数是经过几秒之後才呼叫函式.

实际上来说, 当一个物件呼叫 call_out() 时, 它就被加到一个延迟呼叫的物件表中, 此表中记有延迟呼叫的总延迟时间, 和欲呼叫的函式名称. driver 的每一个周期, 就会进行倒数, 直到呼叫函式的时间. 时间一到, driver 把此物件从延迟呼叫表上删除, 并执行呼叫延迟呼叫函式, 传入原本延迟呼叫函式所指定的参数.

如果你想在一个延迟呼叫执行前将其删除, 你需要用 remove_call_out() 外部函式, 传入延迟呼叫的函式名称. driver 会删除下一次延迟呼叫的这个函式. 这表示如果同一个函式有一个以上的延迟呼叫, 就会出现模拟两可的情况.

要让一个延迟呼叫循环执行, 你必须在你延迟呼叫的函式中再使用 call_out() 外部函式, 因为 driver 执行完延迟呼叫後, 会自动把函式从延迟呼叫表中删除.
举例:

```
void foo() { call_out("hello", 10); }

void hello() { call_out("hello", 10); }
```

在 foo() 第一次被呼叫後, 每 10 秒呼叫 hello() 一次. 在此有几件事要注意. 第一, 你必须要小心, 确定你的延迟呼叫不会造成任何不正确的递回方式. 第二, 比较 set_heart_beat() 和 call_out() 所做的事有何不同.

set_heart_beat():

- a) 将 this_object() 加在心跳物件列表中
- b) 每一次 driver 周期呼叫 this_object() 中的 heart_beat() 函式

call_out():

- a) 将 this_object() 、this_object() 中的函式名称、延迟时间、一组参数, 加在延迟呼叫函式的列表上
- b) 指定名称的函式只呼叫一次, 在延迟一段指定的时间後, 执行此次呼叫

你可以看到, 延迟呼叫的 (a) 部分有很庞大的记忆总量 (memory overhead), 而心跳的 (b) 部分则有更庞大的 CPU 总量, 假设延迟呼叫的延迟时间要比一次 driver 周期来得长.

很明显, 你不会执行延迟一秒的延迟呼叫, 否则你会拖垮两者. 同样, 你也不希望应该使用比一秒钟长的延迟呼叫周期来达成的功能出现在心跳中. 我个人听过一种论点, 认为你应该多使用延迟呼叫. 我最常听到的是, 单一呼叫或比十秒长的周期最好使用延迟呼叫. 十秒以内的周期性呼叫, 你最好使用心跳. 我并不知道这种说法是否正确, 但是我也不认为遵照这种作法会造成任何损害.

---

## 2.5 总结

---

基於更深入了解 LPC, 和了解 driver 和 mudlib 间的互动. 你现在应该知道 driver 执行函式的顺序, 并了解有关 this_player()、add_action()、move_object() 外部函式和 init() 区域函式更多的细节. 另外, 根据以往你从基础 LPC 课本学得的知识, 本章以 driver 如何控制延迟呼叫和心跳来介绍它们. 你现在应该对延迟呼叫和心跳有基本的认识, 并可以在你的程式码中实验一下.



---

译者: Spock of the Final Frontier 98.Jul.22.

---

回到上一页

# 中阶 LPC

Descartes of Borg
November 1993

## 第叁章: 复杂资料型态

### 3.1 简单的资料型态

在基础 LPC 课本里, 你学到常见的基本 LPC 资料型态: 整数 (int)、字串 (string)、物件 (object) 、无传回值 (void). 重要的是, 你学到很多运算式(operation) 和函式 (function) 会因为运算不同的变数资料型态而有不同的行为. 如果你用错资料型态, 有的运算子 (operator) 和函式会给你错误讯息. 例如: "a" + "b" 处理起来就跟 1 + 1 不同. "a" + "b" 把 "b" 加在 "a" 的後面, 得到 "ab". 另一方面, 1 + 1 你不会得到 11, 你会得到你所期望的 2.

我把这些资料型态归类为简单资料型态, 因为它们基本到无法拆成更小的资料型态元件. 物件资料型态是个例外, 但是你实际上也没办法知道它由什麽元素组成. 所以我把它归类为简单资料型态.

本章介绍复杂资料型态的概念, 它是由许多简单资料型态单元所组成的. LPC 有两种常见的复杂资料型态, 两种都属於阵列. 第一种, 传统的阵列 (array), 以连续的各个元素储存数值, 并以数字代表所储存的值在第几号元素 (element) 中. 第二种是称为映射 (mapping) 的关联性阵列 (associative array). 映射把一些数值结合起来, 让资料处理起来更接近一般人的习性.

### 3.2 数值: NULL (虚无) 和 0

深入了解阵列以前, 第一个要先彻底了解的观念是 NULL 的观念和 0 的观念. 在 LPC 中, 一个虚无值 (null value) 由整数 0 代表之. 虽然整数 0 和 NULL 常常随意转换, 在你进入复杂资料型态的领域时, 这种情况常会导致莫大的困扰. 你可能在使用字串时, 已经碰过此种困扰.

0 对整数来说, 表示你把任何数值加上 0 还是原来的数值. 对任何资料

型态的加法运算来说, ZERO (零值) 对此资料型态来讲, 就是你把任何值加上去都维持原值. 所以: A 加 ZERO 等於 A. A 是一个已定资料型态的值, 而且 ZERO 是该资料型态的零值. 这并不算是任何一种正式的数学定义. 虽然还是有一种定义, 但是我不是数学家, 所以我也不知道它的名词是什麽. 总之对整数来说, 0 是零值, 因为 1 + 0 等於 1.

另一方面来说, NULL 表示没有任何值或没有意义. LPC driver 如果能了解 NULL 在该处的意义, 就把 NULL 解释成整数 0. 除了整数的加法以外, 加上 NULL 会导致错误. NULL 产生错误的原因是, 把那些资料型态加上其他没有值的资料型态没有意义.

从另一个观点来看, 我们知道 "a" 加上什麽值结果会得到 "a", 所以我们可以得出字串的零值. 答案不是 0, 而是 "". 对整数来说, NULL 和 0 可以互换是因为 0 代表整数资料型态没有其值. 这种可互换性对其他的资料型态并不适用, 因为其他资料型态的零值并不代表没有其值. 换句话说, "" 表示一个没有长度的字串, 而它与 0 完全不一样.

当你第一次宣告任何型态的变数, 它都没有值. 除了整数以外, 在执行任何运算之前都需要经过初始化. 通常, 全域变数在 create() 函式中初始化, 而区域变数在区域函数的开端指定某些值, 通常是该资料型态的零值. 举例来说, 在以下的程式码中, 我想要作出一个由随机单字组成的字串:

```
string build_nonsense() {      string str;
    int i;

    str = ""; /* 在此, str 以字串的零值初始化 */
    for(i=0; i<6; i++) {
        switch(random(3)+1) {
            case 1: str += "bing"; break;
            case 2: str += "borg"; break;
            case 3: str += "foo"; break;
        }
        if(i==5) str += ".\n";
        else str += " ";
    }
    return capitalize(str);
}
```

如果我们没有对 str 初始化, 尝试把一个字串加上零值会导致错误. 不过, 在此段程式码中将 str 以字串的零值 "" 初始化. 之後, 程式进入一个有六次周期的回圈, 每次把字串加上叁个单字的其中一个. 除了最後一个单字之外, 每个单字後面均加上一个空白字元. 此函式最後离开回圈, 把这个无意义的字串转换成大写, 然後结束.

---

## 3.3 LPC 的阵列 (array)

---

字串是 LPC 一种强大的复杂资料型态, 让你在一个单一变数中存取多个值. 举例来说, Nightmare mud 中, 玩家交易时使用多种货币. 但是, 其中只有五种货币是硬货币 (hard currency). 在此, 硬货币随时可以兑换成其他种类的硬货币, 但是软货币 (soft currency) 只能购买之, 不能出售. 在银行里, 有一张硬货币表让银行老板知道哪种货币属於硬货币. 使用简单资料型态, 每次处理货币兑换交易时, 我们必须要执行以下难看的运算:

```
int exchange(string str) {
    string from, to;
    int amt;

    if(!str) return 0;
    if(sscanf(str, "%d %s for %s", amt, from, to) != 3)
      return 0;
    if(from != "platinum" && from != "gold" && from !=
      "silver" &&
      from != "electrum" && from != "copper") {
        notify_fail("我们不接受软货币 !\n");
        return 0;
    }
    ...
}
```

以五种硬货币来说, 我们有一个相当简单的例子. 全部只需要两行的程式码, 用於 if 叙述中过滤不接受兑换的货币种类. 但是, 如果你必须检查所有游戏中不能使用的货币种类, 怎麽办？ 游戏中可能有 100 种; 你想写一百条 if 叙述 ?如果你想在硬货币表上加上一种新的货币呢？ 这表示, 你必须把游戏中每一项检查硬货币的 if 子句加入新的部分. 阵列让你简易地存取一组相关的资料, 让你每次执行运算时, 不用分别处理

每一个值.

一个阵列常数看起来大概像这样:

```
({ "platinum", "gold", "silver", "electrum", "copper" })
```

这是一个字串阵列. 阵列中个别的资料值称为元素 (element), 或是有时候称为成员 (member). 在程式码里, 作为常数的字串前後以 "" 表示, 阵列常数前後以 ({ }) 表示, 阵列中个别的元素以 , (逗号) 分开.

你可以使用任何简单的或复杂的 LPC 资料型态阵列. 由不同种类的值所组成的阵列称作混合 (mixed) 型态阵列. 在多数的 LPC driver 中, 你使用一种 C 语言的阵列语法来宣告阵列. 这种语法常常困扰撰写 LPC 程式的人, 因为这种语法在 C 中的意义并不能转用到 LPC 中. 无论如何, 如果我们想用一个字串型态的阵列, 我们要用以下的方式宣告它:

```
string *arr;
```

换句话说, 阵列中包含的元素, 其资料型态之後跟着一个空白字元和一个星号.不过请你记住, 新宣告的字串阵列, 其宣告时里头是 NULL 值.

---

## 3.4 使用阵列

---

你应该了解如何宣告并认识程式码中的阵列. 要了解它们在程式码中如何运作, 让我们回顾一下前面银行的程式码, 这次我们用阵列:

```
string *hard_currencies;

int exchange(string str) {
    string from, to;
    int amt;

    if(!str) return 0;
    if(sscanf(str, "%d %s for %s", amt, from, to) != 3)
 return 0;
    if(member_array(from, hard_currencies) == -1) {
        notify_fail("我们不接受软货币 !\n");
        return 0;
```

```
    }
    ...
}
```

这段程式码假设 hard_currencies 是一个全域变数, 并且在 create() 中初始化:

```
    hard_currencies = ({ "platinum", "gold", "electrum", "silver",
"copper" });
```

最佳的做法是把硬货币在标头档 (header file) 中定义为 #define, 让所有的物件都能使用之, 不过 #define 在以後的章节会提到.

一旦你知道 member_array() 外部函式的功能後, 这种方式就比较容易读懂, 也比较容易撰写. 实际上, 你大概已经猜到 member_array() 外部函式的功能: 它告诉你一个指定的值是否在某个阵列中. 此处特别是指, 我们想知道玩家想卖出的货币是否为 hard_currencies 阵列中的元素. 你可能会感到混淆的是, member_array() 不只告诉我们特定值是否为阵列中的元素, 实际上还告诉我们阵列中的哪一个元素是此值.

它要怎麼告诉你是哪个元素？ 如果你把阵列变数当作是拥有一个数字, 就比较容易了解它. 对上面的参数举例来说, 我们假设 hard_currencies 拥有 179000 的值. 这个值告诉 driver 要到哪里寻找 hard_currencies 所代表的阵列. 所以, hard_currencies 指向一个可以找到阵列值的地方. 当有人谈到阵列的第一个元素时, 它们希望该元素位於 179000. 当一个物件需要阵列第二个元素的值时, 它就找 179000 + 一个值, 然後 179000 加上两个值就是第叁个, 以此类推. 我们因此可以藉由阵列元素的索引来存取个别的阵列元素, 索引就是在阵列起点之後第几个值, 而我们在阵列中找寻数值. 对 hard_currencies 阵列来说:

"platinum" 索引为 0.
"gold" 索引为 1.
"electrum" 索引为 2.
"silver" 索引为 3.
"copper" 索引为 4.

如果在阵列中有此种货币, member_array() 传回其元素的索引, 如果阵列中没有则传回 0. 要参考一个阵列中的单独元素时, 你要照着以下的方式使用之:

阵列名称[索引号]

范例:

hard_currencies[3]

hard_currencies[3] 会是 "silver".

所以, 你现在应该知道阵列以全体或个别元素出现的方式. 全体而言, 你用它的名称参考 (reference) 之, 而一个阵列常数前後以 ({ }) 围住, 并且用 , (逗号) 分隔其元素. 对个别的元素而言, 你用阵列名称跟着前後加上 [] 的索引号码来参考阵列变数, 而对阵列常数来说, 你可以如同相同型态的简单资料型态常数般参考之.

整个阵列:

变数: arr
常数: ({ "platinum", "gold", "electrum", "silver", "copper" })

阵列中个别的元素:

变数: arr[2]
常数: "electrum"

你可以将这些参考的方式, 用於你以前习惯其他资料型态的方法. 你可以指定其值、将其值用於运算式中、将其值当成参数传入函式中、用其值当作传回值. 请记得一件很重要的事, 当你单独处理一个元素时, 单独的元素本身不是阵列 (除非你处理的是阵列的阵列). 在上述的范例中, 单独的元素是字串. 所以:

```
str = arr[3] + " and " + arr[1];
```

会造出一个字串等於 "silver and gold". 虽然这看起来很简单, 很多刚开始接触阵列的人试着在阵列中加入新元素时, 就遇到麻烦. 当你处理整个阵列, 并想要加入新元素时, 你必须用另一个阵列加上去.

注意以下的例子:

```
string str1, str2;
```

```
string *arr;

str1 = "hi";
str2 = "bye";

/* str1 + str2 等於 "hibye" */

arr = ({ str1 }) + ({ str2 });

/* arr 等於 ({ str1, str2 }) */
```

更深入以前, 我必须说明这个制作阵列的例子是极为恐怖的方法. 你应该这样来设定阵列: arr = ({ str1, str2 }). 不过, 这个例子的重点是, 你必须以同样的资料型态进行加法. 如果你试着把一个元素以其资料型态加入一个阵列, 你会得到错误. 你必须将它视为一个只有单一元素的阵列处理之.

---

## 3.5 映射 (mapping)

---

LPMud 中, 一个最重要的进步是创立了映射资料型态. 大家亦称它为关联性阵列. 实际上来说, 一个阵列让你不用像阵列般使用数字索引一个值. 映射让你使用实际上对你有意义的值当作其值的索引, 比较像一个相关的资料库 (relational database).

在一个有五个元素的阵列中, 你个别使用它们 0 到 4 的整数索引存取这些值. 想像一下, 再回到钱币的范例中. 玩家有不同数量、不同种类的钱币. 在玩家物件中, 你需要一个方法储存这些钱币的种类, 并把该种货币与玩家有多少数量连结起来. 对阵列来说, 最好的方法就是储存一个表示钱币种类的字串阵列, 和另一个整数阵列代表有多少钱. 这样会产生一段吃光 CPU 的难看程式码:

```
int query_money(string type) {
    int i;

    i = member_array(type, currencies);
    if(i>-1 && i < sizeof(amounts))  /* sizeof 外部函式传回元素的总数 */
        return amounts[i];
```

```
    else return 0;
}
```

这是一个简单的查询函式. 接下来看一个加法函式:

```
void add_money(string type, int amt) {
    string *tmp1;
    int * tmp2;
    int i, x, j, maxj;

    i = member_array(type, currencies);
    if(i >= sizeof(amounts)) /* 错误的资料, 我们用了一个烂方法 */
        return;
    else if(i== -1) {
        currencies += ({ type });
        amounts += ({ amt });
        return;
    }
    else {
        amounts[i] += amt;
        if(amounts[i] < 1) {
            tmp1 = allocate(sizeof(currencies)-1);
            tmp2 = allocate(sizeof(amounts)-1);
            for(j=0, x =0, maxj=sizeof(tmp1); j < maxj;
              j++) {
                if(j==i) x = 1;
                tmp1[j] = currencies[j+x];
                tmp2[j] = amounts[j+x];
            }
            currencies = tmp1;
            amounts = tmp2;
        }
    }
}
```

这实在是一些很烂的程式码, 只为了增加钱这种简单的概念. 首先, 我们要得知玩家有哪些种类的钱币, 如果有, 它是货币阵列中的哪一个元素. 之後, 我们必须检查更动过之货币资料是否完整. 如果在货币阵列中, 货

币种类的索引大於钱币数量阵列的元素总数, 则我们就出了问题. 因为这两个阵列之间仅靠索引连结其关系. 只要我们知道资料正确无误, 如果玩家手上目前没有该种货币, 我们仅把这种货币当作新的元素加入货币阵列, 并把其数量也当作新元素加入数量阵列. 最後, 如果玩家手上持有该种货币, 我们就把其数量加在数量阵列中相对的索引上. 如果钱币数量小於 1, 表示用完该种货币, 我们想把该种货币从记忆体中清除之.

从一个阵列中减去一个阵列不是一件简单的事. 举个例子, 下面的结果:

```
string *arr;

arr = ({ "a", "b", "a" });
arr -= ({ arr[2] });
```

你认为 arr 最後的值是多少？ 唔, 它是:

```
({ "b", "a" })
```

从原来的阵列减去 arr[2] 并不会从该阵列中除去第叁个元素. 反之, 它从该阵列减去其第叁个元素的值. 而阵列的减法是把该阵列中第一次出现的该值删除之. 既然我们不想被迫去计算该元素在阵列中是否唯一, 我们就被迫要翻几个 斗以从两个阵列中同时除去正确的元素. 如此才能保持两个阵列索引的关联性.

映射提供了一个比较好的方式. 它们让你直接把钱币种类和其总数连结在一起. 有些人认为映射就相当於, 一种不限制你只能用整数当索引的阵列. 事实上, 映射是一种彻底不同的概念, 用於储存多个集团资讯. 阵列强迫你选择一种对机器才有意义的索引, 该索引用於寻找正确资料位置之用. 这种索引告诉机器在首值之後第几个元素才是你想要找的值. 而映射, 你可以选择对你有意义的索引, 不用担心机器要怎麼去寻找和储存它.

以下是映射的格式:

常数:

```
整个: ([ 索引:值, 索引:值 ]) 例: ([ "gold":10, "silver":20 ])
元素: 10
```

变数值:

整个: map  (map 是映射变数的名称)
元素: map["gold"]

所以现在我的货币函式看起来像:

```
int query_money(string type) { return money[type]; }

void add_money(string type, int amt) {
   if(!money[type]) money[type] = amt;
   else money[type] += amt;
   if(money[type] < 1)
     map_delete(money, type);          /* 用於 MudOS */
          ...或...
        money = m_delete(money, type)  /* 用於 LPMud 3.* 衍生版本 */
          ... 或...
       m_delete(money, type);    /* 用於 LPMud 3.* 衍生版本 */
}
```

请先注意, 从一个映射中清除一个映射元素的外部函式, 每种 driver 都不同. 查询你的 driver 文件说明, 以得知适当的外部函式名称及语法.

你可以马上看到, 你不需要检查你资料的完整性, 因为你想得知的两个值密不可分地结合在一起. 另外, 删除无用的值只需要一个简单的外部函式呼叫, 不用一个繁杂而耗费 CPU  的回圈. 最後, 查询的函式只需要一行 return 指令.

使用映射以前, 你必须宣告并将其初始化.

宣告看来如下:

```
mapping map;
```

而通常初始化看来如下:

```
map = ([]);
map = allocate_mapping(10)   ...OR...   map = m_allocate(10);
map = ([ "gold": 20, "silver": 15 ]);
```

跟其他的资料型态一样, 它们通常的运算也有其规则定义, 像是加法和减法:

```
([ "gold":20, "silver":30 ]) + ([ "electrum":5 ])
```

得到:

```
(["gold":20, "silver":30, "electrum":5])
```

虽然我的示范显示出映射有个顺序, 但是实际上, 映射在储存元素时, 不保证会遵照其顺序. 所以, 最好别比较两个映射是否相等.

---

## 3.6 总结

---

映射和阵列可以依照你的需求, 要有多复杂就有多复杂. 你可以造出一个阵列的映射的阵列. 这种东西可以宣告如下:

```
mapping *map_of_arrs;
```

它看起来像:

```
({ ([ ind1: ({valA1, valA2}), ind2: ({valB1, valB2}) ]),
 ([ indX: ({valX1,valX2}) ]) })
```

映射可以使用任何一种资料型态作为索引, 包括物件. 映射索引常常称作关键 (key), 是来自资料库的名词. 你随时要谨记在心, 对於任何非整数的资料型态而言, 作一般像是加法或减法的运算使用之前, 你必须先将其变数初始化. 虽然利用映射和阵列撰写 LPC 程式变得简单又方便, 没有正确地将其初始化所产生的错误, 常常把刚接触这种资料型态的新手逼疯. 我敢说大家最常碰到映射和阵列的错误, 是以下叁者之一:

```
Indexing on illegal type.
Illegal index.
Bad argument 1 to (+ += - -=) /* 看你最喜欢哪一种运算 */
```

第一个和第叁个几乎都是因为出问题的阵列或映射没有正确初始化. 第二种错误讯息通常是当你试着使用一个已初始化过的阵列中所没有的索引. 另外, 对阵列来说, 刚接触阵列的人常得到第叁种错误讯息, 因为他们常试着将一个单独的元素加入一个阵列, 把初始的阵列与单一的元

素值相加, 而没有把一个含有该单一元素的阵列与初始的阵列相加. 请记住, 只能把阵列加上阵列.

行文至此, 你应该觉得能自在地使用映射和阵列. 刚开始使用它们时, 应会碰上以上的错误讯息. 使用映射成功的关键, 在於除去这些错误讯息, 并找出你程式设计上, 何处使你试着使用没有初始化的映射和阵列. 最後, 回到最基本的房间
程式码, 并看看像是 set_exits() 之类的函式 (或在你的 mudlib 上相当的函式). 它有可能使用映射. 在某些情况下, 它会使用阵列以保持与 mudlib.h 的相容性.



---

译者: Spock of the Final Fronier 98.Jul.24.

---

回到上一页

---

# 中阶 LPC

Descartes of Borg
November 1993

## 第四章: LPC 前编译器 (pre-compiler)

### 4.1 回顾

上一章的份量相当重, 所以我现在的步调会放慢一些, 藉由 LPC 前编译器这个简单的课题, 让你能消化并使用映射和阵列. 不过在此, 你应该相当了解 driver 如何与 mudlib 互动, 并能撰写呼叫延迟呼叫和心跳的物件. 附带一提, 你应该撰写一些使用映射和阵列的简单物件, 注意这些资料型态如何在物件中使用. 开始阅读实际的 mudlib 程式码是个不错的主意, 这样能让你制作你自己的 mud. 看看你自己是否了解你的 mudlib 房间和怪物程式码其中的每一件事. 对你不懂的事, 就询问你 mud 中负责回答创作人程式码问题的人.

前编译器实际上有点误导人, 因为 LPC 码永远不会真正编译过. 虽然这一点随着新的 LPC driver 原型而渐渐改变, LPC driver 解译创作人所写的程式码, 而非编译为二进位格式. 虽然如此, LPC 前编译器的功能仍然表现得比较像是编译语言的前编译器, 其指令甚至在 driver 开始看物件码之前就已解译.

### 4.2 前编译器指令

如果你不知道什麼是前编译器, 你不用担心. 对 LPC 而言, 它基本上是在 driver 开始解译 LPC 码, 以让你执行档案中整段程式码的动作之前的一个程序. 因为程式码还未解译, 前编译器程序在档案以物件存在之前、检查任何 LPC 函式和指令之前执行. 所以前编译器在档案层次上工作, 表示它并不会处理任何在继承档案中的程式码.

前编译器在送给它的档案中寻找前编译器指令. 这些档案中的小指令只对前编译器有意义, 并不算是 LPC 语言的一部份. 一个前编译器指令是在档案中任何以 # 号开头的一行. 前编译器指令通常用於制造一个档案看起来的最终程式码. 最常见的前编译器指令是:

#define #undefine
#include
#ifdef
#ifndef
#if
#elseif
#else
#endif
#pragma

mud 里大多数的区域码撰写人并不使用 #define 和 #include 指令. 其他你常见的指令即使你从未用过, 你也大概知道它们的意义.

第一对指令是:

#define
#undefine

#define 指令设定一组字元, 这组字元在程式码中的任何地方都会在前编译器处理时段换成它们所定义的东西.
举例:

```
#define OB_USER "/std/user"
```

这个指令让前编译器寻找整个档案中是否有 OB_USER. 任何有 OB_USER 的地方, 它就换成 "/std/user". 注意, OB_USER 在程式码中并不算是变数. LPC 解译器永远不会看到 OB_USER 的标签. 前面已经说过, 前编译器是在程式码解译之前的一段过程. 所以你所写的:

```
#define OB_USER "/std/user"

void create() {
   if(!file_exists(OB_USER+".c")) write("Merde! No user file!");
   else write("Good! User file still exists!");
}
```

到了 LPC 解译器的手上就变成:

```
void create() {
   if(!file_exists("/std/user"+".c")) write("Merde! No user file!");
```

```
    else write("Good! User file still exists!");
}
```

只要放个 #define, 它就会将定义的标签换成标签後面的任何东西. 你也可以把#define 用於一种特殊的情况, 标签後面不跟着任何值. 这种情形称为二进位定义 (binary definition). 举个例子:

```
#define __NIGHTMARE
```

出现在 Nightmare Mudlib 的组态档 (config) 中. 这样让前编译器测试一些东西, 我们在本章稍後会说明.

其他你常用的前编译器指令是 #include. 正如其名字所暗示的, #include 在前编译时将其他档案的内容放入该指令出现的地方. 专为其他档案纳入而制作的档案常称为标头档 (header file). 它们有时候含有一些东西被很多档案共用, 像是 #define 指令和函式宣告. 标头档传统的档案延伸名是 .h .

include 指令的语法有两种:

```
#include <filename>
#include "filename"
```

如果你用档案的绝对名称, 则你用哪一种语法都无所谓. 档案名称前後使用什麽符号决定前编译器如何寻找标头档. 用 <> 围住的档案, 前编译器首先寻找系统 include 目录. 用 "" 围住的档案, 前编译器开始从前编译器正在处理的档案所在之目录找起. 不然在放弃之前, 前编译器会寻找系统 include 目录和该档案所在的目录. 使用的语法决定了寻找的顺序.

最简单前编译器指令是 #pragma 指令. 我怀疑你大概从未使用过. 基本上, 你在 #pragma 指令之後跟着对 driver 有意义的一些关键字. 我唯一见过的关键字是 strict_types, 它让 driver 知道你希望这个档案以严格资料型态解译之. 我怀疑你会需要使用这种指令, 而且你可能从未看过它. 我在此介绍它, 只是因为当你看到它时, 不会让你认为它实际上不具有任何意义.

最後一组前编译器指令是条件前编译器指令 (conditional pre-compiler directives) . 它们让你在一个运算式为真值时, 以一种方式前编译一个档案, 运算式为伪值时, 以另一种方式前编译该档案. 这是让程式码在不同

mudlib 之间具有移植性 (portable) 最方便的方法, 举例来说, 因为在 MudOS mud 的程式码中放入 m_delete() 外部函式会导致错误, 所以你大概会照着以下撰写:

```
#ifdef MUDOS
    map_delete(map, key);
#else
    map = m_delete(map, key);
#endif
```

经过前编译器处理之後, 解译器会看到:
在 MudOS mud 中:

```
    map_delete(map, key);
```

其他的 mud:

```
    map = m_delete(map, key);
```

解译器永远看不到会产生错误的函式呼叫.

请注意, 我前面用於说明二进位定义的例子. 二进位定义让你对解译器传入一些程式码, 基於其他条件下, 你所使用的 driver 或 mudlib 为何.

---

## 4.3 总结

---

前编译器是在你程式之间维持模组性的有用工具. 当你有易受影响而改变的值, 而此值在你的档案中普遍使用, 你可以在标头档使用 #define 叙述将它们全部置换之, 这样一来你以後需要改变这些值时, 只需要更改 #define 指令. 在此最好的例子是 money.h , 它包含这个指令:

```
#define HARD_CURRENCIES ({ "gold", "platinum", "silver",
"electrum", "copper" })
```

如果你想加上新的硬货币, 你只需要更改这个指令, 就能更新所有需要硬货币为何的档案.

LPC 前编译器也让你撰写不用随 mudlib 和 driver 而改写的可携性程式码. 最後, 你应该小心, 前编译器只接受以 carriage return 结束的一行字.

如果你要撰写一个多行的前编译器指令, 你必须在未结束的一行末尾加上反斜线 (\).



---

译者: Spock of the Final Frontier 98.Jul.26.

---

回到上一页

---

# 中阶 LPC

Descartes of Borg
November 1993

## 第五章: 高级的字串处理

### 5.1 字串是什麽

基础 LPC 课本教你字串是简单资料型态. LPC 一般来说也这样处理字串. 不过, 在底下的 driver 程式是以 C 写成的, 它没有字串资料型态. driver 实际上视字串为复杂资料型态, 由字元的阵列所组成 ---- 一种简单的 C 资料型态. LPC 在另一方面来说, 并不认识字元资料型态 (可能有一两种 driver 认得字元资料型态, 但是一般上来说不认得) . 其结果是, 你可以对字串作一些类似阵列的处理, 而其他的 LPC 资料型态则否.

你第一个该学与字串有关的外部函式是 strlen(). 这个外部函式传回一个 LPC 字串中, 以字元为单位的长度. 就从这个外部函式的行为来说, 你可以看到 driver 视字串由更小的元素所组成, 并以此处理之. 在本章之中, 你将学到如何以更基础的字元和子字串层次处理字串.

### 5.2 字串是字元阵列

你可以对阵列作的事, 几乎都可以用於字串, 除了在字元基础上指定其值以外. 最基本的是, 你实际上可以在字元前後加上 ' (单引号) 将它当作字元常数. 所以 'a' 和 "a" 在 LPC 中是完全不一样的东西. 'a' 表示是一个字元, 不能用於指定叙述或其他的运算式中 (比较两值的式子除外). 另一方面, "a" 是由单一字元所组成的字串. 你可以加减其他的字串, 并指定它为变数值.

对字串变数来说, 你可以存取单独的字元跟字元常数作比较. 其语法与阵列相同. 换句话说, 以下叙述:

```
if(str[2] == 'a')
```

是一个有效的 LPC 叙述, 将 str 的第二个字元与 'a' 字元作比较. 你必须非常小心, 你不会把阵列元素与字元相比较, 也不会把字串的字元与字串相比较.

LPC 也让你使用范围运算子 (range operator) .. 一起存取多个字元:

```
if(str[0..1] == "ab")
```

换句话讲, 你可以看 str 字串中第 0 到 1 个字元是什麽. 如同阵列, 你必须小心使用索引或范围运算子, 才不会试着参考比最後一个索引还大的索引数. 这样会导致错误.

现在你可以看到字串和阵列之间的几处相似点:

- 1) 两者你都可以藉由索引存取个别的元素.
  - a) 字串个别的元素是字元.
  - b) 阵列个别的元素符合阵列的资料型态.

- 2) 你可以运算一个范围之内的值.
  - a) 例: "abcdef"[1..3] 是 "bcd" 字串
  - b) 例: ({ 1, 2, 3, 4, 5 })[1..3] 整数阵列 ({ 2, 3, 4 })

当然, 你应该记住基本上的相异点: 字串不是由更基本的 LPC 资料型态所组成. 换句话说, 你没办法将值指定给字串中单独的字元.

---

## 5.3 sscanf() 外部函式

---

不使用 sscanf(), 你在 LPC 中就无法更有效处理字串. 没有它, 你就只能处理传给命令函式之命令叙述的整个字串. 换句话讲, 你没办法处理一个像 "give sword to leo" 的命令, 因为你没有方法分析 "sword to leo" 的成分. 像这种使用多个参数的命令, 它们使用 sscanf() 外部函式让命令更接近英文.

大部分的人都觉得 sscanf() 的说明文件相当难懂. 这个函式并不算是非常符合说明文件中的格式. 如同前述, 这函式用於读取字串, 并分析出有用的成分. 技术上来说, 它读取一个字串, 并分析成一个或一个以上的各种型态之变数. 举个

例子:

```
int give(string str) {
    string what, whom;

    if(!str) return notify_fail("Give what to whom?\n");
    if(sscanf(str, "%s to %s", what, whom) != 2)
      return notify_fail("Give what to whom?\n");
    ... 其馀的 give 程式码 ...
}
```

sscanf() 外部函式需要叁个以上的参数. 第一个参数是你想分析的字串. 第二个参数称为控制字串. 控制字串是一个模型, 表示原来所写的字串格式为何, 它该如何分析. 其馀的参数是变数, 你会由控制字串指定值给它们.

控制字串由叁种不同的元素组成:

- 1) 常数
- 2) 被分析的变数参数
- 3) 要丢弃的变数

在 sscanf() 之中你变数参数的数目必须与控制字串中第二种元素的数目相等. 在上述的例子中, 控制字串是 "%s to %s", 是叁个元素的控制字串, 由一个常数部分 (" to ") 和两个被分析的变数参数 ("%s") 组成. 在此没有要丢弃的变数.

控制字串基本上指出函式应该在 str 字串中寻找 " to ". 在此常数之前不管是什麽东西, 会以字串型态放在第一个变数参数中. 同理, 常数後面的任何东西, 会放在第二个.

变数元素以 % 符号跟着一个解释码表示. 如果变数元素要丢弃, % 符号之後跟着 * 号, 再跟着解释变数的码. 常见的变数元素解释码是 s 表示字串, 和 d 表示整数. 另外, 你的 mudlib 可能支援其他的转换码, 像是 f 表示浮点数. 所以在上述的两个例子中, 控制字串中的 %s 指出原来字串中, 不管什麽东西出现在对应的位置上, 就会以字串被分析成新的变数.

来一个简单的练习. 你要怎麽把字串 "145" 转成一个整数 ?

答案:

```
int x;
sscanf("145", "%d", x);
```

sscanf() 执行之後, x 会等於整数 145.

无论何时, 你使用控制字串分析一个字串, 函式会寻找原来字串中第一次出现第一个常数的地方. 举个例, 如果你的字串是 "magic attack 100", 并撰写了以下的程式码:

```
int improve(string str) {
    string skill;
    int x;

    if(sscanf(str, "%s %d", skill, x) != 2) return 0;
    ...
}
```

你会发现你得到 sscanf() 错误的传回值 (稍後再多讨论传回值) . 控制字串 "%s %d", 是由被分析的两个变数和一个常数组成的. 常数是 " ". 所以函式寻找原字串中第一次出现 " " 的地方, 把 " " 之前的任何东西放入 skill, 并试着把 " " 之後的任何东西放入 x. 这样一来, 把 "magic attack 100" 分成 "magic" 和 "attack 100" 两个部分. 但是函式没办法把 "attack 100" 变成一个整数, 所以它传回 1, 表示有一个变数值成功分析出来 ("magic" 转 skill).

也许你已经从上面的例子中猜到, 但是 sscanf() 外部函式传回一个整数, 是从原字串成功分析出来的变数值个数. 这里有些传回值的例子让你看看:

| | |
|---|---|
| sscanf("swo  rd descartes", "%s to %s", str1, str2) | 传回: 0 |
| sscanf("swo  rd descartes", "%s %s", str1, str2) | 传回: 2 |
| sscanf("200 gold to descartes", "%d %s to %s", x, str1, str2) | 传回: 3 |
| sscanf("200 gold to descartes", "%d %*s to %s", x, str1) | 传回: 2 |

x 是一个整数, 而 str1 和 str2 是字串.

## 5.4 总结

LPC 字串可以视为字元的阵列, 但是你要牢记的是, LPC 并没有字元资料型态 (绝大多数, 但不是所有的 driver 皆是). 既然字元不是一种真正的 LPC 资料型态, 你就无法像其他资料型态一样, 处理一个 LPC 字串中单独的字元. 注意, 虽然字串和阵列之间的相似关系可以让你比较容易了解字串的范围运算子和索引的概念, 两者仍有不同之处.

虽然除了 sscanf() 之外, 高级的字串处理仍牵涉到其他的外部函式, 它们却不常需要用到. 你应该阅读你 mud 中这些外部函式的 man 或 help 档案: explode() 、implode() 、replace_string()、sprintf(). 这些都是非常有价值的工具, 尤其是你想在 mudlib 层次上撰写程式码之时.



译者: Spock of the Final Frontier 98.Jul.26.

回到上一页

# 中阶 LPC

Descartes of Borg
November 1993

## 第六章: 中级继承 (inheritance)

### 6.1 基础继承

在基础 LPC 课本中, 你学到 mudlib 如何藉由继承维持 mud 物件之间的一致性. 继承让 mud 管理人撰写所有的 mudlib 物件, 或某一种的 mudlib 物件都必须拥有的基本函式, 让你可以专心创作使物件独树一格的函式. 当你建造一个房间、武器、怪物时, 你使用一套早已替你写好的的函式, 并将它们让你的物件继承之. 以此方法, 所有 mud 中的物件可以依靠别的物件表现某种方式的行为. 举个例, 玩家物件实际上依靠所有房间物件其中称为 query_long() 的一个函式以得知房间的叙述. 继承让你不用担心 query_long() 长得如何.

当然, 这份课本会试着超越继承的基本知识, 让程式撰写人更了解 LPC 程式设计中, 继承如何运作. 目前还不需要深入高级区域程式码撰写人 / 初级 mudlib 程式撰写人要知道的细节. 本章会试着详细解释, 你继承一个物件时所发生的事.

### 6.2 复制 (cloning) 与继承

当一个档案第一次以一个物件被参考 (相对於读取档案的内容) , 游戏试着将档案载入记忆体, 并创造一个物件. 如果该物件成功载入记忆体, 它就成为主本 (master copy) . 物件的主本可被复制, 但是不用作实际上的游戏物件. 主本用於支援游戏中任何的复制物件.

主本是 mud LPC 程式撰写争辩的源头之一, 也就是要复制它还是继承它. 对房间来说就没有问题, 因为在游戏中每个房间物件应该只有一份. 所以你一般使用继承来创造房间. 很多 mud 管理人, 包括我自己在内, 鼓励创作人复制标准的怪物物件, 并从房间物件中设定之, 而不是让怪物分为单独的档案, 并继承标准怪物物件.

如同我前述的部分, 每次一个档案被参考, 用於创造一个物件时, 一份主本就会被载入记忆体. 像是你做以下的事:

```
void reset() {     object ob;
   ob = new("/std/monster");
     /* clone_object("/std/monster") some places */
   ob->set_name("foo monster");
   ... 其馀的怪物设定程式码, 之後再将怪物搬入房间中 ...
}
```

driver 会寻找是否有一个称为 "/std/monster" 的主物件. 如果没有, 它就创造一个. 如果存在, 或已被创造出来, driver 就创造一个称为 "/std/monster#<编号>" 的复制物件. 如果此时是第一次参考 "/std/monster" , 结果会创造两个物件: 主物件和复制物件.

另一方面, 让我们假设你在一个继承 "/std/monster" 的特殊怪物档案中的 create() 里面, 已经做好所有的设定. 不从你房间复制标准怪物物件, 而你复制你自己的怪物档案. 如果标准怪物尚未载入, 因为你的怪物继承它, 所以载入之. 另外, 你档案的一个主本也被载入记忆体. 最後, 创造出一份你怪物的复制, 并搬入你的房间. 总共游戏中增加了叁个物件. 注意, 你无法轻易地使用主本做到这些. 举例来说, 如果你想做:

```
"/wizards/descartes/my_monster"->move(this_object());
```

而非

```
new("/wizards/descartes/my_monster")->move(this_object());
```

你会无法修改 "my_monster.c" 并更新它, 因为更新 (update) 指令摧毁一个物件现存的主版本. 在某些 mudlib 中, 它也载入新版本到记忆体中. 想像一下, 玩家在战斗中杀得如火如荼的时候, 因为你更新档案让怪物消失无踪！ 此时他们的脸色可不好看.

所以当你只是计划要复制时, 复制是一个有用的工具. 如果你对怪物并没有做什麽特殊的事, 又不能藉由几个外界呼叫 (call other) 做到, 那你可以避免载入许多无用的主物件而节省了你 mud 的资源. 不过, 如果你计画要对一个物件增加一些功能 (撰写你自己的函式) 或是如果你有一个单独的设定多次重复使用 (你有一队完全一样的半兽人守卫, 所以你撰写一个特别的半兽人档案并复制之), 继承就相当有用.

当 A 物件和 B 物件继承 C 物件, 叁个物件全都有自己的一套资料, 而由 C 物件共享一套函式定义. 另外, A 和 B 在它们个别的程式码中会有自己的函式定义. 因为本章馀下的部分都需要范例说明, 我们使用以下的程式码. 在此别因为一些看起来没有意义的程式码而困扰.

C 物件

```
private string name, cap_name, short, long;
private int setup;

void set_name(string str);
nomask string query_name();
private int query_setup();
static void unsetup();
void set_short(string str);
string query_short();
void set_long(string str);
string query_long();


void set_name(string str) {
    if(!query_setup()) {
        name = str;
    setup = 1;
}

nomask string query_name() { return name; }

private query_setup() { return setup; }

static void unsetup() { setup = 0; }

string query_cap_name() {
    return (name ? capitalize(name) : ""); }
}

void set_short(string str) { short = str; }
```

```
string query_short() { return short; }

void set_long(string str) { long = str; }

string query_long() { return str; }

void create() { seteuid(getuid()); }
```

B 物件

```
inherit "/std/objectc";

private int wc;

void set_wc(int wc);
int query_wc();
int wieldweapon(string str);

void create() { ::create(); }

void init() {
    if(environment(this_object()) == this_player())
      add_action("wieldweapon", "wield");
}

void set_wc(int x) { wc = x; }

int query_wc() { return wc; }

int wieldweapon(string str) {
    ... code for wielding the weapon ...
}
```

A 物件

```
inherit "/std/objectc";

int ghost;

void create() { ::create(); }
```

```
void change_name(string str) {
    if(!((int)this_object()->is_player())) unsetup();
    set_name(str);
}

string query_cap_name() {
    if(ghost) return "A ghost";
    else return ::query_cap_name();
}
```

你可以看到, C 物件被 A 物件和 B 物件继承. C 物件代表的是一个相当简化的基本物件, 而 B 也是相当简化的武器, A 是简化的活物件. 虽然我们有叁个物件使用这些函式, 每一个函式在记忆体中只维持一份. 当然, 从 C 物件而来的变数在记忆体中有叁份, 而 A 物件和 B 物件各有一份变数在记忆体中. 每一个物件有自己的资料.

---

## 6.4 函式和变数标签 (label)

---

注意, 以上的许多函式是以本文和基础课本中还未介绍过的标签处理之, 这些标签就是 static (静态)、private (私有)、nomask (不可遮盖) . 这些标签定义一个物件的资料和函式拥有特殊的特权. 你至今所使用的函式, 其预设的标签是 public (公共). 只有某些 driver 预设如此, 有的 driver 并不支援标签.

一个公共变数是物件宣告它之後, 其继承树之下的所有物件皆可使用之. 在 C 物件中的公共物件可以被 A 物件与 B 物件存取之. 同样, 公共函式在物件宣告它以後, 可以被继承树之下的所有物件呼叫之.

相对於公共的是私有. 一个私有变数或函式只能由宣告它的物件内部参考之. 如果 A 物件或 B 物件试着参考 C 物件中的任何私有变数, 就会导致错误, 因为这些变数它们根本看不到, 或说因为它们有私有标签, 无法被继承物件使用.
不过, 函式提供一个变数所没有的独特挑战. LPC 外部物件有能力藉由外界呼叫 (call other) 呼叫其他物件中的函式. 而私有标签无法防止外界呼叫.

要防止外界呼叫, 函式要使用静态标签. 一个静态函式只能由完整的物件内部或 driver 呼叫之. 我所谓的完整物件就是 A 物件可以呼叫它所

继承 C 物件中的函式. 静态标签只防止外部的外界呼叫. 另外, this_object()->foo() 就算有静态标签, 也视为内部呼叫.

既然变数无法由外部参考, 它们就不需要一个同效的标签. 某几行程式里, 有人决定要捣蛋, 并对变数使用静态标签以造成完全不同的意义. 更令人发狂的是, 这标签在 C 程式语言里头一点意义也没有. 一个静态变数无法经由 save_object() 外部函式储存, 也无法由 restore_object() 还原. 自己试试.

一般来说, 在一个公共函式中有一个私有变数是个很好的练习, 使用 query_*() 函式读取继承变数的值, 并使用 set_*()、add_*() 和其他此类的函式改变这些值. 在撰写区域程式码时, 这实际上并不需要担心太多. 实际上的情形是, 撰写区域程式码并不需要本章所谈的任何东西. 不过, 要成为真正优秀的区域程式码撰写人, 你要有能力阅读 mudlib 程式码. 而 mudlib 程式码到处都是这些标签. 所以你应该练习这些标签, 直到你可以阅读程式码, 并了解它为什麽要以这种方式撰写, 还有它对继承这些程式码的物件有何意义.

最後一个标签是不可遮盖, 因为继承的特性允许你重写早已定义的函式, 而不可遮盖的标签防止此情形发生. 举例来说, 你可以看到上述的 A 物件重写 query_cap_name() 函式. 重写一个函式称为僭越 (override) 该函式. 最常见的函式僭越就像这样, 当我们的物件 (A 物件) 因为特殊的条件情况, 需要在特定情形下处理函式呼叫. 在 C 物件中, 因为了 A 物件可能是鬼魂而放入测试的程式码, 是一件很蠢的事. 所以, 我们在 A 物件中僭越 query_cap_name(), 测试该物件是否为鬼魂. 如果是, 我们改变其他物件询问其名字时所发生的事. 如果不是鬼魂, 我们想回到普通的物件行为. 所以我们使用范围解析运算子 (scope resolution operator, ::) 呼叫继承版本的 query_cap_name() 函式, 并传回它的值.

一个不可遮盖函式无法经由继承或投影 (shadow) 僭越之. 投影是一种反向继承, 将在高级 LPC 课本中详细介绍. 在上述的范例中, A 物件和 B 物件 (实际上, 其他任何物件也不行) 无法僭越 query_name(). 因为我们想让 query_name() 作为物件唯一的鉴识函式, 我们不想让别人透过投影或继承欺骗我们. 所以此函式有不可遮盖标签.

---

## 6.5 总结

---

透过继承, 一个程式撰写人可以使用定义在其他物件中的函式, 以避免

产生一堆相似而重复的物件, 并提高 mudlib 物件与物件行为的一致性. LPC 继承允许物件拥有极大的特权, 定义它们的资料如何被外部物件和继承它们的物件存取之. 资料的安全性由 nomask、private、static 这些标签维持之.

另外, 一个程式码撰写人能藉由僭越, 改变非防护函式的功能. 甚至在僭越一个函式的过程中, 一个物件可以透过范围解析运算子存取原来的函式.



---

译者: Spock of the Final Frontier 98.Jul.28.

---

回到上一页

---

# 中阶 LPC

Descartes of Borg
November 1993

## 第七章: 除错

### 7.1 错误的种类

至今, 你大概已经到处碰过各式各样的错误. 一般上, 你可能看到的错误有叁种: 编译时段错误 (compile time error) 、 执行时段错误 (run time error) 、 故障的程式码 (malfunctioning code). 在大多数的 mud 中, 你会找到一个私人的档案, 里头记录着你的编译时段错误. 对大多数人来说, 你可以在你的家 (home) 目录找到名叫 "log" 或 ".log" 的档案, 或在 "/log" 目录找到以你的名字命名的档案. 另外, mud 执行时, 会维持一份执行时段错误的纪录. 而此档案也在 "/log" 目录中. 对 MudOS mud 来说, 它叫做 "debug.log". 其他的 mud 中, 称为不同的名字, 像是 "lpmud.log". 如果你还不知道编译时段和执行时段错误纪录在哪里, 请询问你的系统管理者.

编译时段错误是 driver 试着载入一个物件到记忆体的时候发生的错误. 如果此时它看不懂你写的东西, 它会无法把物件载入记忆体, 并在你私人的错误纪录档中记录为什麼它无法载入该物件. 最普遍的编译时段错误是打字错误、 遗漏或多加 () , {}, [], ""、 没有正确宣告物件所使用的函式和变数.

执行时段错误是一个在记忆体中的物件, 当它执行某段叙述时所发生的错误. 举例来说, driver 不可能知道任何情况下, "x/y" 是否有效. 实际上, 它是一个有效的 LPC 运算式. 但是, 如果 y 的值为 0, 则会发生执行时段错误, 因为你不能除以 0. 当 driver 执行一个函式时碰上错误, 它放弃执行函式并纪录在游戏执行时段错误纪录档中. 如果有定义、 如果玩家是创作者, driver 也会对 this_player() 显示错误讯息, 不然就只对玩家显示 "什麼 ?". 大多数导致执行时段错误的原因, 是不正确的值和试着执行没有定义运算资料型态的运算式.

不过, 最狡猾的错误种类, 就是故障的程式码. 这些错误不会纪录下来, 因为 driver 永远不可能知道有地方出错. 简单地说, 这种错误就是你认

为程式码做的是一件事, 但是实际上它做的是另一件事. 常遇到这种错误的人, 一定会认定是 mudlib 或 driver 的错误. 每个人都制造过各式各样的错误, 而更常见的不是程式码不按照它该运作的的方式工作, 而是你错读它.

---

## 7.2 修正编译时段错误

---

编译时段错误是最常见以及最容易修正的错误. 新手程式撰写人常常因为一些怪异的错误讯息, 而感到挫折. 虽然如此, 只要一个人变得习惯於他们 driver 产生的错误讯息, 修正编译时段错误就成了例行公事.

在你的错误纪录中, driver 会告诉你错误的种类, 还有它最後在第几行注意到该错误. 注意, 这不表示此行一定是错误实际发生的地方. 除了打字错误, 最常见的编译时段错误是遗漏或多加各式括号和引号: (), [], {}, "". 这种是最常困扰新手程式撰写人的错误, 因为 driver 不会注意到遗漏或多加的部分, 直到稍後出问题为止. 以下是范例:

```
1 int test(string str) { 2    int x;
3    for(x =0; x<10; x++)
4       write(x+"\n");
5    }
6    write("Done.\n");
7 }
```

看你想做的是什麼, 此处实际上的错误在第叁行 (表示你遗漏了一个 () 或第五行 (表示你多加一个 }) . 但是, driver 会回报它在第六行找到一个错误. 实际的 driver 讯息每种 driver 可能都不一样, 但是不管是哪一种 driver, 你会看到第六行产生一个错误. 因为第五行的 } 会解释为 test() 函式结束. 在第六行, driver 看见你有一个 write() 出现在函式定义之外, 所以回报为错误. 一般来说, driver 也会继续回报它在第七行找到一个多加 } 的错误.

修正这种错误的秘诀在於程式撰写风格. 将结束的 } 与该子句开头的 { 垂直对齐, 在你除错时, 会让你看到你哪里遗漏它们. 同样, 当你使用多组括号时, 像这样用空白将各组分开:

```
if( (x=sizeof(who=users()) > ( (y+z)/(a-b) + (-(random(7))) ) )
```

你可以看到, for() 叙述的括号与其馀的叙述以空白隔开. 另外, 个别的子群也用空白隔开, 让它们在产生错误时易於找出.

一旦你拥有帮助你找出错误的程式撰写风格, 你就会学到哪一种错误讯息倾向於指出哪一种错误. 修正此种错误时, 你会检查出问题的那一行之前与之後的程式码. 大多数的情况下, 你会直接找到错误.

另一种普遍的编译时段错误是 driver 回报一个不明的 identifier. 一般来说, 打字错误和错误宣告变数导致此种错误. 幸运的是, 错误纪录档中几乎都能告诉你错误所发生的实际位置. 所以修正此种错误时, 进入编辑程式并找到出问题的该行. 如果该问题出在变数上而不是打字错误, 请确定你正确地宣告该变数. 另一方面, 如果是打字错误, 就改正它 !

但是, 小心一件事, 这种错误有时候会与遗漏括号的错误结合在一起. 在这种情形下, 不明 identifier 的问题常常是误报. driver 误读 {} 或其他东西, 而导致变数宣告混淆. 因此在烦恼此种错误困扰之前, 请确定已修正所有其他的编译时段错误.

与前述的错误同一级的是普通的语法错误. 当 driver 无法了解你写的东西时, 它就产生此种错误. 这又常是打字错误引起的, 却也是因为不了解某些特徵正确的语法所致, 像是把 for() 叙述写成这样:

```
for(x=0, x<10, x++)
```

如果你像这样的错误, 却不是语法错误, 试着重新检查错误发生的叙述中, 语法是否正确.

---

## 7.3 修正执行时段错误

---

执行时段错误比起编译时段错误要复杂得多. 幸运的是, 这些错误都有纪录, 但是许多创作人并不了解, 或是他们不知道纪录在哪里. 执行时段错误的纪录一般也纪录得比编译时段错误详细, 也就是你可以从它开始到它出错之处, 追踪执行程序的过程. 所以你可以利用纪录档, 使用前编译器叙述 (precompiler statement) 设置除错陷阱 (debugging trap). 但是, 执行时段错误常肇因於复杂的程式撰写技巧, 而初学者并不使用这些技巧. 这表示你一般会碰上比简单的编译时段错误还要复杂的错误.

执行时段错误几乎都是肇因於使用错误的 LPC 资料型态. 最常见的是, 试着用 NULL 值的物件变数做外界呼叫, 索引指向 NULL 值的映射、

阵列、字串变数, 或函式传入错误的参数. 我们看一个 Nightmare 真实的执行时段错误:

```
Bad argument 1 to explode()
程式: bin/system/_grep.c, 物件: bin/system/_grep 第 32 行
'       cmd_hook' in '        std/living.c' ('       std/user#4002') 第 83 行
'       cmd_grep' in '  bin/system/_grep.c' ('    bin/system/_grep') 第 32
行

Bad argument 2 to message()
程式: adm/obj/simul_efun.c, 物件: adm/obj/simul_efun 第 34 行
'       cmd_hook' in '        std/living.c' ('      std/user#4957') 第 83 行
'       cmd_look' in '  bin/mortal/_look.c' ('   bin/mortal/_look') 第 23 行
' examine_object' in '  bin/mortal/_look.c' ('   bin/mortal/_look') 第 78
行
'         write' in 'adm/obj/simul_efun.c' (' adm/obj/simul_efun') 第 34
行

Bad argument 1 to call_other()
程式: bin/system/_clone.c, 物件: bin/system/_clone 第 25 行
'       cmd_hook' in '        std/living.c' ('      std/user#3734') 第 83 行
'      cmd_clone' in ' bin/system/_clone.c' ('  bin/system/_clone') 第 25
行

Illegal index
程式: std/monster.c, 物件: wizards/zaknaifen/spy#7205 第 76 行
'     heart_beat' in '       std/monster.c' ('wizards/zaknaifen/spy#7205') 第
76 行
```

除了最後一个以外, 所有的错误, 都对一个函式传入一个错误的参数. 第一个错误, 是对 explode() 传入错误的第一个参数. explode() 外部函式要一个字串当作第一个参数. 修正这类型的错误时, 我们会到 /bin/system/_grep.c 的第 32 行检查第一个传入参数到底其资料型态为何. 在此情况下, 传入的值应是字串.

如果因为某些原因, 我实际上传入其他的东西, 我在此只要确定传入字串就能修正错误. 但是在此情况要复杂得多. 我需要追踪传入 explode() 的变数值为何, 我才能知道传入 explode() 外部函式的值到底是什麽.

出问题的那行是:

```
borg[files[i]] = regexp(explode(read_file(files[i]), "\n"), exp);
```

files 是一个字串阵列, i 是整数, borg 是映射. 所以很明显, 我们需要找出 read_file(file[i]) 的值到底是什麽. 好, read_file() 这个外部函式传回一个字串, 除非该档案根本不存在, 或是该物件没有权限读取该档案, 或是该档案是个空的档案, 这些情形都会导致此函式传回 NULL. 很明显, 我们的问题是这些情形的其中一种. 要找出是哪一种, 我们要看 file[i].

检查程式码, 这个档案阵列透过 get_dir() 外部函式取得它的值. 如果该物件有权限读取此目录, get_dir() 就传回目录中所有的档案. 所以问题不在於权限不足或档案不存在. 导致这个错误的档案一定是空的. 而且事实上, 这就是导致错误的原因. 要修正此错误, 我们要透过 filter_array() 外部函式传入档案, 确定只有档案大小大於 0 的档案可以读入阵列.

修正执行时段错误的关键在於, 了解有问题的所有变数值在产生错误之时, 它们确实的值为何. 你阅读你的执行时段错误纪录时, 小心地经由错误发生的档案分辨物件. 举个例子, 上面的索引错误是物件 /wizard/zaknaifen/spy 产生, 但是错误发生在执行它所继承的 /std/monster.c 函式.

---

## 7.4 故障的程式码

---

你所遇到最阴险的问题, 就是你程式码的行为不是预期中的行为. 物件顺利载入, 没有产生任何执行时段错误, 但是事情就是不对劲. 既然 driver 不可能认出这种错误, 就没有任何纪录. 所以你需要一行接一行浏览整个程式码, 并搞清楚到底发生了什麽事.

第一步: 找出你已知能顺利执行的最後一行程式码.
第二步: 找出你已知开始出错的第一行程式码.
第叁步: 从已知顺利执行的地方到第一个出错的地方, 检查程式码的流程.

常常, 这些问题出现於你使用 if() 叙述没有料到所有的可能情形. 举个例:

```
int cmd(string tmp) {
    if(stringp(tmp)) return do_a()
```

```
    else if(intp(tmp)) return do_b()
    return 1;
}
```

在此段程式码中, 我们发现它编译和执行起来没有问题. 问题是它执行起来完全没作用. 我们确定 cmd() 函式已经执行, 所以我们可以从此着手. 我们也知道实际上 cmd() 传回 1, 因为我们输入此命令时, 没看到 "什麼 ?" . 马上, 我们可以看到因为某些原因, tmp 变数有字串或整数以外的值. 就此得到的答案是, 我们输入的命令没有参数, 所以 tmp 是 NULL , 并让所有的测试条件失败.

上面的例子相当简单, 几近於愚蠢. 但是, 它让你知道在修正故障的程式码时, 如何检查程式码的流程. 其他的工具能协助你除错. 最重要的工具就是使用前编译器来除错. 以前面的例子来说, 我们有一个子句检查传入 cmd() 的整数. 我们输入 "cmd 10" 时, 我们希望执行 do_b() . 进入回圈之前, 我们需要看 tmp 的值为何:

```
#define DEBUG

int cmd(string tmp) {
#ifdef DEBUG
    write(tmp);
#endif
    if(stringp(tmp)) return do_a();
    else if(intp(tmp)) return do_b();
    else return 1;
}
```

在我们输入命令之後, 立刻就知道 tmp 的值是 "10" . 回头看程式码, 我们会怪自己愚蠢, 忘了我们把 tmp 当整数使用之前, 必须要用 sscanf() 把命令参数转换成整数.

---

## 7.5 总结

---

修正任何 LPC 问题的关键是, 永远要知道你程式码中任何一步的变数值为何. LPC 的执行降到变数值改变这种最简单的层级上, 所以程式码载入记忆体时, 不正确的值导致错误发生. 如果你遇到函式有不正确的参数时, 常常是你对一个函式传入 NULL 值的参数. 这情形常发生於物

件, 因为大家常常会做以下的事:

1) 使用设定在一个物件中的值, 而该物件已经被摧毁.
2) 使用 this_player() 的传回值, 而根本没有 this_player().
3) 使用 this_object() 的传回值, 而 this_object() 刚好已被摧毁.

另外, 大家会常常遇上不合法的索引 (illegal indexing) 或索引指向不合法的型态 (indexing on illegal types). 最常见的是因为有问题的映射或阵列没有初始化, 所以无法索引之. 关键在於了解出问题的地方, 其阵列和映射的完整值. 另外, 注意索引编号是否比阵列的长度还大.

最後, 使用前编译器暂时扔出或扔进显示变数值的程式码. 前编译器让你很容易地删掉除错程式码. 你只需要在除错完毕之後, 删除 DEBUG 定义.



---

译者: Spock of the Final Frontier 98.Jul.29.

---

回到上一页

---