ZMUD462--宝典

Zmud4.62北大侠客行mud提供下载pkuxkx.net
例如：别名（aliases），动作（actions），宏（macros），快捷键（keys），
按钮（buttons），脚本（scripts），地图（maps）等，使你在mud中的生活轻松高效。 这里先简单介绍一下zMUD的特性

一、宏（Macro keys） 定义快捷键以便快速执行一个或一系列命令。

二、别名（Aliases） 将一系列命令保存为一个较短的命令。

三、触发（Triggers）根据从mud获得的信息决定执行的命令。

四、变量（Variables）

五、函数（Functions）

六、按钮（Buttons）在窗口中建一排你自己定义的按钮。

七、路径（Paths）在mud中记录走过的路径，你可以沿着记录下的路径倒走，
也可以编辑记录的路径。

八、地图（Mapper）

九、多角色（Multiple Chars）同时登录多个角色，一个命令可以发往不同的窗口
或发往所有的窗口。

十、防止向mud送出太多相同的字串的 Spam protection 。

十一、全部的 ANSI color 支持，颜色可以被定制。

十三、可代替Telnet

十四、多媒体功能允许你播放 sounds, MIDI, movies, 等等。

十五、脚本（Scripts）将命令保存在文本文件中。

十六、角色数据库（Character Database）保存你所有的mud角色，便于快速连接。

十七、Tab completion 允许你键入一条长字符串的头几个字母然后用Tab
键补全。 参见#ta命令。

十八、历史（History）相当于doskey。（还记得doskey吗？）

十九、Timer 内建的计时器用于定时向MUD中送出命令。

二十、Status line 可定制的状态行可以显示变量和触发。

二十一、颜色、字体、声音、特殊字符都能被修改并保存。

二十二、配置文件保存所有的配置 (aliases, macros, etc)。 一个配置文件可以用于几个不同的角色。

[number]

Syntax: #number command

这个命令向MUD重复发送number次命令，number必须是一个常数，如果要使用变量，请使用loop命令。当前已经重复的计数保存在系统变量%repeatnum中。

实例：

#4 fight bing %i

命令将被解释成{fight bing 1;fight bing 2;...;fight bing 4}发往MUD10

---

abort

Syntax: #AB

停止分析处理当前的命令行中#ab以后的部分

实例：

get all corpse;#ABORT;split
在这个例子中，split将被忽略

---

action

Syntax: #AC pattern command [classname]
Related: #TRIGGER #T+ #T- #IGNORE

这是zMUD中最有用的特性之一。当从MUD中收到一条匹配的文本时，预先定义的命令将被执行。这条命令的第一个参数是被匹配的文本，如果文本中包含空格，你需要将它用括号{}括起来。pattern中能够包含特殊的pattern matching symbols 和通配符。第二个参数是将要被执行的命令，超过一个单词的命令也需要使用括号{}。第三个参数用于给不同的触发动作命名分类，便于管理。高级的触发设置必须使用参数（Preferences）对话框，在对话框中，你可以决定触发的动作是紧跟在匹配的文本之后还是在新的一行中执行。

一个简单的例子：

#AC {你买下一件藤甲} {wear jia}

这样，无论何时只要你买了藤甲，立刻可以自动穿上。

自动登录的例子：

#AC {^您的英文名字：} {river}
#AC {^请输入相应密码：} {12345}

然后在参数对话框中关闭 Trigger on Newline 设置并且打开 Trigger on Prompt 这样名字和密码将紧接着提示行输入而不会等待换行。注意，字符 ^ 将强制从一行的头部开始匹配。

触发中的参数

#AC {^You get (%d) coins} {chat I get %1coin } rich

当你看到一行“You get [number] coins”时，其中的数值将被存放在变量%1中，其后执行的命令可以使用这个数值。在这个例子中用到了类（class）名rich，这样你就能够使用t+和t-命令来打开或关闭这个触发。

---

add

Syntax: #AD variable amount

这个例子允许你做一个简单的算术计算。amount可以是数值或变量。如果要做减法，

可以使用负值。复杂一些的算术计算，请使用 #math 命令。

实例：

#AD moves 1
给变量@moves加一

#ACTION {You get (%d) coins} {#AD gold %1}
当捡到钱时，把捡钱的数量加到 @gold 变量中

---

alarm

Syntax: #ALA timepattern command
Related: #TRIGGER

建立一个基于时间的触发，而不是根据从MUD中获得的信息触发。timepattern可以是特定的时间或包括通配符。如果在minus前加上（-），则表示连线时间而不是实际时间。
timepattern 的格式表示成 hours:minutes:seconds ，其中的小时和分钟是可选项，如果不指明，则假定小时和分钟被通配符*取代。通配符*可以代替任何数值，而*10可以匹配10、20、30等等。你也可以指定几个数值用（|）来分隔。Finally, you can put parenthesis around the wildcards to save the values matched to the %1..%9 parameters。（这句话我不懂，谁来帮我？）

实例：

#ALARM -30:00 {save}
这里的 hour 没有指定，所以默认为*。 因此，这个例子将在连线后每隔30分钟执行 save 命令。

#ALARM 3:00:00 {gossip Why arent you sleeping?}
命令将在凌晨3点触发。

#ALARM -59:(55|56|57|58|59) {#SHOW 60-%1}
这个命令在你连线时每个小时的最后5秒在屏幕上显示 5 4 3 2 1。

---

alias

Syntax: #AL [aliasname] [string]
Related: #VARIABLE

保存一个或一组命令到一个较短的别名中。当执行别名时，命令中的变量能够被扩展。如果需要延迟变量扩展，可以使用两个变量标志（%%）。

如果不带参数执行命令alias，所有的别名将被列出在输出窗口。如果在命令alias中指定别名，被指定的别名将被显示。

别名能够用<TAB>键扩展。在命令行中键入一个别名并按下<TAB>键，命令行中的别名将被替换成别名中包含的命令。

如果在定义别名时使用参数（%1，%2，...），在命令行中紧随别名的文本将取代这些参数。特殊的参数%-1到%-99表示-n之后的所有文本。（使用参数的例子）

实例

#AL fs {fill waterskin}

当执行fs时，fill waterskin将被送往MUD。

#AL fs {fill @container}

在别名中使用变量，变量可以在购买容器时用触发赋值。

#AL make {#ALIAS %1 {cast %1 %%1}}

在这个例子中，执行make heal将发出#ALIAS heal {cast heal %1}命令，这样就建立了
一个新的别名heal。

---

all

Syntax: #ALL command

发送一个命令到所有角色的窗口

实例：

#ALL quit
发送 quit 命令到所有激活的角色窗口。

---

backup

Syntax: #BA
Related: #PATH #RETRACE

从当前正在记录的路径中删除最后一步

实例：

如果当前路径是 .nsew 那麼执行 #BA 将使其成为 .nse 。如果当前路径是 .n4s 则执行 #BA 后将成为 .n3s 。

---

beep

Syntax: #BEEP [value]
Related: #PLAY

在pc机喇叭中发声，value 对应相应的 windows 事件。

实例：

#BEEP 16
播放windows 事件 16 的声音。

#BEEP;#WAIT 500;#BEEP
beep两次，中间间隔0.5秒。

---

button

Syntax: #BU number

触发一个自定义按钮，编号（从1至16）决定触发的按钮。number 可以是一个变量。

实例：

#BU 1

触发第一个按钮，效果相当于在屏幕上按下这个按钮

---

c+

Syntax: #C+ [name]
Related: #C-

开始获取文本到窗口name中，如果省略name，文本将送入command editor，（如果editor中的capturing设置可用）。如果在preferences对话框中选中Capture Commands设置，键入的命令也将被送往这个窗口。

例子：

#c+ temp

开始拷贝所有从MUD中获得的文本到名为temp的窗口中

---

c-

Syntax: #C-
Related: #C+

停止获取文本到另一个窗口

---

capture

Syntax: #CAP [number] [name]
Related: Editor window

获取最后number行送到另一个窗口。如果省略number，则只获取最后一行。如果number是-1，所有的行将被拷贝。如果name省略，发送的目标将是command editor window。

例子

#tr {咖啡告诉你：救命！} {#cap tell}

将tell的内容存入名为tell的窗口，避免忽略。

---

case

Syntax: #CA index command1 [commandn]

允许从命令列表中选择一个命令执行。index参数决定执行哪一个命令。如果index
大于列表中的命令数，将从第一个开始重新选择。例如，列表中有4个命令，
而index=6，则第二个命令将被执行。你也可以用变量%random随机的选择需执行的命令。

实例：

#CASE 2 {first command} {second command} {third command}

执行第二个命令

#CASE @joincmd {join} {rescue}

如果变量@join是奇数，执行join，偶数执行rescue。

#CASE %random {Hello} {Hi there} {Hiya} {Hi}

随机选择问候语。

---

character

Syntax: #CH
Related: #HOST #PW

从Charater atabase中返回当前角色的名字

---
colse

Syntax: #CL filenum
Related: #FILE

根据给定的文件号关闭文件，文件必须是已经用#file命令打开的。

实例：

#CLOSE 1

关闭一号文件

---

clr

清除屏幕。如果要清空scrollback buffer并收回内存，请使用菜单命令Empty。

---

color

Syntax: #CO attribute [pattern]
Related: #HIGHLIGHT

如果省略pattern参数，这个命令将改变最后一行的颜色。颜色属性见下表。
如果包括pattern参数，将建立一个颜色触发，引起触发的条件除了字符对应外还需要有相同的颜色。

Color values:

black 0
blue 1
green 2
cyan 3
red 4
magenta 5
brown 6
gray 7
tellow 14
white 15
bold 128

要显示亮色，在基本值上加8，作为背景颜色，需要用基本值乘16。例如：使用红色背景的值是4*16或64。在前景上使用粗体字，需要再加128。这样，在蓝色背景上的粗体白字的值是：128+1*16+15=159。

实例：

#CO red

将最后一行的颜色变成红色

#CO bold,red

将最后一行变成红色粗体

---

connect

Syntax: #CON
Related: #DISCONNECT

断开并重新连接到当前的MUD，与菜单命令File/Reconnect 相同

---

cr

Syntax: #CR

送一个空行到MUD

---

cw

Syntax: #CW color
Related: #COLOR

在一次成功的触发之后，改变引起触发的内容的颜色

实例：

#TRIGGER {告诉你} {#CW red}

当有人tell 你时，显示较为醒目的颜色

---

dde

Syntax: #DDE server topic macro

这个命令允许你通过动态数据交换使用外部程序。zMUD也有一些内建的函数用于DDE：

%dde(server,topic,item)

%ddepoke(server,topic,item,value)

%ddemacro(server,topic,macro)

如果你使用%ddeopen(server,topic)函数打开DDE连接，在其他函数中不再需要指明server和topic，这个函数是全局性的，对于zMUD的任何窗口均有效。结束DDE连接时，使用%ddeclose()函数。

zMUD也拥有自己的DDE服务，server名是zmud，topic也是zmud，item为data。

实例：

#DDE NETSCAPE WWW_OpenURL {http://www.zg169.net/~~czmud/index.html}

使用netscape打开zMUD从入门到精通主页，因为~zMUD的特殊字符，所以需要再加一个~写成~~。

#DDE ZMUD ZMUD {chat* bye;quit}

使用DDE向MUD发出命令

%dde(Excel,TEST.XLS,R1C1)

使用excel装入test.xls并返回单元R1C1的值。

%ddepoke(Excel,TEST.XLS,R1C1,@tank)

向test.xls的R1C1单元写入变量@tank的值

谁能用DDE写一个自动对诗的机器人？

---

default

Syntax: #DE [special-char-string]
Related: #NODEF

保存你当前使用的特殊字符并恢复系统默认的特殊字符，在使用现成的命令脚本时，这个命令可以保证脚本中的标准特殊字符被正确识别。使用#NODEF命令将恢复你自定义的特殊字符。

zMUD中使用的特殊字符有

1 Command Char # 将紧随的字串解释为命令
2 Separator Char ; 分隔多个命令
3 Variable Char @ 用于扩展变量的值
4 History Char ! 调用命令缓冲区中的命令
5 Parameter Char % 指出系统变量和函数
6 Movement Char . 调用路径变量或函数
7 Focus Char : 向指定窗口发命令
8 Quote Char ~ 紧随其后的特殊字符当作普通字符处理
9 Must be a space at the end *? *fool

实例：

#DEF

保存你当前使用的特殊字符并恢复系统默认的特殊字符

#DEF {xx$xxxxx }

保存你当前使用的特殊字符并回复默认的特殊字符，然后将@改为$。

---

disconnect

Syntax: #DI
Related: #CONNECT

断开当前的连接

实例：

#TRIGGER {你已陷入半昏迷状态} {#DI}

断线就不会被杀死吗？我没试过，hehe。

ECHO

Syntax: #EC string
Related: #SAY

在当前窗口中显示字串，类似与say命令。除了用于离线测试trigger似乎没有别的用处。

实例：

ERASE

Syntax: #ERA filenum
Related: #FILE

从硬盘中删除用#file打开的文件

实例：

#FILE 1 old.log
#ERA 1

删除old.log文件，需要先用#file命令打开。

EXEC

Syntax: #EXEC command

执行一个命令，命令中可以包括变量

实例：

#TRIGGER {^咖啡告诉你(*)} {#EXEC %1}

执行咖啡的指令。（我可没那麼傻）

FILE

Syntax: #FI number name
Related: #READ #WRITE

打开文件准备读写。zMUD可以打开10个文件，文件号1～5打开文本文件，能够按顺序读或追加，文件号6～10打开记录文件用于随机读写。如果nuber已经用于打开的文件，则前一个文件将被关闭。打开的文件只能位于zmud.exe所在的目录且不能是EXE,HLP和MUD文件。

实例：

#FILE 1 test.txt

打开text.txt作为文件1。

FIND

Syntax: #FIN

找出在当前地图上的定位，通过look命令用当前房间的描述比较地图数据定位匹配的房间，与automapper的菜单命令find作用相同。

FORALL

Syntax:#FO list command

逐条列出字串列表内容并执行命令。列表使用|分隔，逐个赋值给变量%i，

并执行命令。

实例：

list=sys|ups|yahoo

#froall @list {kill %i}

我一定是疯了。

FREEZE

Syntax: #FR [value]

切分屏幕查看以前的显示，value=0表示恢复屏幕。这条命令相当于Ctrl-z或单击右边的滚动条。
（所以没什么用）

GAG

Syntax: #GA [pattern]
Related: #UNGAG

如果pattern省略，命令将从屏幕上删除最后一行。如果包括pattern，将删除所有匹配行，相当于#ACTION pattern '#GAG'。

实例：

#ga 咖啡

这叫眼不见为净。

HELP

Syntax: #HE [command]

不带参数相当与从菜单中选择help，否则显示指定命令的参数。

H+

Syntax: #H+
Related: #H-

从命令缓冲区中取得下一条命令,只在使用过#h-命令后有效，
相当于向下的箭头键。

H-

Syntax: #H-
Related: #H+

从命令缓冲区中取得上一条命令，相当于向上的箭头键。

HISTORY

Syntax: #HIS

显示命令缓冲区中的命令，每条显示的命令都带有行号，可以用
!加行号来执行某条命令。!!执行最新的命令。!＋字母将执行最
近一个以!后字母开头的命令。命令缓冲区的大小默认为20，
可由参数对话框改变。

你也可以用左键单击命令输入行左边的三角以弹出交互命令缓冲区
对话框来选择命令，单击将命令送至命令输入行以供编辑，双击将
命令直接送往MUD。

<TAB>键可以配合命令缓冲区的使用，如果你用!加数字或字母，
再按下<TAB>键，命令将拷入命令输入行以供编辑。

实例：

#Hl

显示最后20条命令

!!

执行最后一条命令

!3

执行命令缓冲区中的最后一条命令

!k

执行最后一条以k开头的命令

!k<TAB>

将最后一条以k开头的命令拷入命令输入行以供编辑

HIGHLIGHT

Syntax: #Hl [pattern]
Related: #COLOR

如果省略pattern，最后一行文本将以高亮显示，如果包括pattern，
匹配的文本将以高亮显示。相当于命令#ACTION pattern '#HIGHLIGHT'。

HOST

Syntax: #HO
Related: #CHAR #PW

返回当前连接的主机名

IF

Syntax: #IF expression true-command [false-command]

执行条件分支，如果表达式为真，执行true-command，否则执行
falise-command（可选）。表达式中可以包含变量和运算子。

实例：

#if (@qn>30) {learn_skills}

在战斗中如果潜能达到了30，执行别名learn_skills返回师父处学习。

#if (@hp>10) {exercise @hp} {exert recover}

如果气大于10就修炼内功，否则补气。

#IF (@line =~ "You receive (%d) coins") {split %1}

如果变量@line与pattern "You receive (%d) coins"匹配，
执行{split %1}，判断匹配需要使用“=~”。

IGNORE

Syntax: #IG
Related: #T+ #T-

打开或关闭所有触发，第一次执行#ig 关闭所有触发，第二次执行恢复正常。

INPUT

Syntax: #IN string

将文本放入命令输入行，替换当前内容

实例：

#in get @item

扩展变量item后将命令置入命令输入行

KEY

Syntax: #KE key command

定义按键执行的命令，key值需注明按键的全名，例如：F1，CTRL-A 或 ALT-F2

实例：

＃key F1 eat baozi

定义F1键执行eat baozi 命令

<ALT-D>={drink jiudai}

定义 ALT-D 键执行 drink jiudai 命令

KILLALL

Syntax: #KILLALL

删除所有的别名、宏、触发、<TAB>缩写

LMAP

Syntax: #LM path command
Related: #LOOP

穿越指定的路径并在每个房间执行指定的命令，变量%i将记录房间数

实例：

#LMAP 3sn {#SHOW %roomname(%i)}

从当前的地图定位向南走三步，向北走一步，并显示每个房间的名称

LOAD

Syntax: #LOA filename
Related: #SAVE

装入指定的设置文件，filename中可以使用变量。注意：装入新的
设置文件时，原来的设置文件中的改变不会被保存

实例：

#load dc

装入设置文件dc.mud。（ .mud是默认的扩展名）

<F1>={#load combat};<F2>={#load cocial}

为战争与和平定制的设置文件分别用F1和F2载入

LOOK

Syntax: #LOOK

执行MUD的look命令并重新写入地图的房间数据

LOOP

Syntax: #LOO range command

重复执行由range指定数量的命令，range包括最小值和最大值，
中间用逗号分隔，如果range中只有一个数字，默认的最小值是1，
循环的次数记录在变量%i中，可在命令中调用。

实例：

#LOO 3 north

向MUD中送入north命令3次

#LOO 3,4 {get all from corpse %i}

取出第三和第四具尸体中的所有东西

#LOO @num {eat baozi}

吃包子@num口

LOG

Syntax: #LO [filename]

开始在给定的文件中记录从MUD中获得的文本，如果文件不存在，
则建立文件并开始记录，如果文件已存在，则打开存在的文件并追加记录。
省略参数用于开关记录

实例：

#LO test.txt

开始将MUD的所有输出记入test.txt

#LO

开关记录，如果正在记录，该命令将停止记录，否则开始记录

MAP

Syntax: #MAP direction
Related: #PATH

在当前路径上增加一个方向

#MAP north

如果当前路径是.s，则更新为.sn，同时向北移动一步

MATH

Syntax: #MAT variable expression
Related: #ADD

将表达式的结果赋值给变量。表达式中可以包括数字、逻辑运算和
字符串函数，表达式中包含的变量将被扩展。

实例：

#MATH test (1+3)*4

将计算结果16赋值给变量test

#MATH test2 @test-4

如果@test的值是16，则变量test2将被赋值12

#ALIAS add {#MATH value %1+%2}
add 3 4

执行后变量value的值为7

MARK

Syntax: #MA
Related: #PATH

开始记录新的路径，正在记录的路径将被废弃

MEDIA

Syntax: #ME function
Related: #PLAY

向当前的多媒体设备发送命令，通常在#play之后使用，
function中可以使用变量，可用的function依设备不同而不同值，
一般有：

back 返回一步
close 关闭当前文件
eject 放弃当前设备
next 播放下一曲目
pause 暂停
paly 开始播放
prev 播放前一曲目
resume 恢复暂停的播放
rewind 从头开始播放
step 向前步进
stop 停止播放

实例：

#media next

如果正在播放CD，这个命令将播放下一曲目。

MEMORY

Syntax: #MEM

显示剩余内存

MENU

Syntax: #MEN command

执行一个菜单命令

实例：

#MENU {File|Exit}

执行退出命令

#MENU {Actions|Make Button}

弹出make button对话框

MESSAGE

Syntax: #MES string

在小窗口中显示指定的信息，十秒钟后自动关闭

NAME

Syntax: #NA string

改变当前窗口的名称，默认的窗口名由角色数据库定义。

实例：

#name tank

将当前窗口名改为tank。你可以用tank:command 将命令发往这个窗口。

NOOP

Syntax: #NO

空命令，什么也不做。

NODEF

Syntax: #NODEF

Related: #DEFAULT

恢复被#def命令保存的特殊字符。

NOMAP

Syntax: #NOMAP [pattern]

避免匹配的行干扰地图分析，如果省略参数，前一个引起触发的行将被地图忽略。

实例：

#TRIGGER {闲聊} {#NOMAP}

#NOMAP {闲聊}

以上两条命令作用相同，任何包含“闲聊”的行将不会被地图分析。

OK

Syntax: #OK

Related: #SLOW #STOP

沿定义的路径移动时放慢脚步。

实例：

#TRIGGER {的出口是} {#OK}

当前一步被确认时，允许下一步的行动

PATH

Syntax: #PA [pathname]

Related: #MARK #RETRACE #MAP

省略参数时，显示当前记录的路径。如果包括参数pathname，
当前的路径被存入指定的pathname。方向字符(.)将自动添加在路径的开头。

实例：

#pa

显示当前记录的路径。

#pa magic

将当前路径存入变量 .magic。

PICK

Syntax: #PI val1 [val2 [val3 ...]]

列表中最多可以指定99个值，用户可以选择一个或多个选项作为
命令执行，按<ESC>键放弃。

命令中可以使用p:string 定义string为对话框提示，或用o:1指定
只能选择一个选项，加上*号的val将作为默认选项。可以使用
caption:command方式在对话框中显示命令提示caption。

实例：

#pi {get all from corpse} {get gold from corpse}
{get silver from corpse}

在对话框中显示三条命令供用户选择

#pi {p:选择命令} {o:1} {*get all from corpse}
{get gold from corpse} {get
silver from corpse}

用“选择命令”作为对话框提示，{o:1}指定只能选择一条命令，
*表示get all from corpse 作为默认命令。

#pi {p:选择命令} {o:1} {ALL:get all from corpse}
{GOLD:get gold from corpse}
{SILVER:get silver from corpse}

使用ALL GOLD SILVER作为命令提示

PLAY

Syntax: #PL filename
Related: #MEDIA

播放 wave, midi, avi, cd 或其他媒体，如果包括驱动器号，
则播放音乐CD。

实例：

#play start.wav

播放start.wav 文件

#play d:

播放音乐CD。

sound=ouch.wav
#tr {看起来想杀死你} {#play @sound}

触发声音警告

PROMPT

Syntax: #PR aliasname

弹出对话框显示指定的别名或变量的值。

PW

Syntax: #PW
Related: #CHAR

向MUD发送当前的口令，口令不会在输出屏幕上回显

READ

Syntax: #REA filename
#REA n [rec]

1、逐行读入给定的文件并执行每一行。

2、读入第n个文件的第rec个记录。文件由file命令读入。
如果n是1～5，则给定的文件是文本文件，rec表示行号，
省略rec时，读入下一行。如果n是6～10，表示是一个结构文件，指
定的记录将被读入，省略rec时，读入下一个记录。

实例：

#rea mud.txt

逐行读入并执行mud.txt。

#file 1 mudlist.txt

#read 1 10

读入mudlist.txt的第十行

RECALL

Syntax: #RECALL

使用地图返回teleport的起点

RECORD

Syntax: #REC [aliasname]
Related: #ALIAS

开始或停止记录别名。第一次键入#record时，zMUD开始记录送往MUD的所有命令，你可以再次键入#record查看已记录的命令，结束记录时，键入#rec加指定的别名保存。#rec off 将放弃记录别名。

实例：

#REC
starts recording
n
w
open door
#REC
displays: Current alias: n;w;open door
#REC temple
将命令 n;w;open door 存入别名temple 并停止记录。

RESET

Syntax: #RES n
Related: #FILE

重设文件为初始状态

RETRACE

Syntax: #RE [pathname]
Related: #PATH

沿指定路径倒走，如果pathname省略，则沿当前正在记录的路径返回。

SAY

Syntax: #SA text

在屏幕上回显文本，类似#sh 命令。

SAVE

Syntax: #SAV [filename]
Related: #LOAD

保存当前设置文件

SCROLL

Syntax: #SC pattern [lines]

显示scrollback buffer中所有与pattern相匹配的行，除非指定lines。

SEND

Syntax: #SE filename [prefix] [postfix]

向MUD中发送一个文本文件，并可在每一行中加上前缀或后缀

#SEND notes.txt {tell coffee}

把notes.txt的内容告诉coffee，如果同时使用kill命令，效果更好。*grin

SESSION

Syntax: #SES [character-name|hostname port]

使用指定的角色或主机打开新任务

实例：

#SES river

#SES 168.160.244.39 6666

SHOW

Syntax: #SH text

在屏幕上回显文本，不发往MUD，类似于#say，通常用于测试触发。

SLOW

Syntax: #SL path
Related: #STEP #STOP #OK

沿路径慢走，前一步得到确认后再执行下一步。#ok命令用于确认
行动完成，#stop用于放弃继续前进，#step继续被#stop放弃的行程。

实例：

#SL .n2es

north 命令首先被发往MUD，然后等待确认以执行下一个命令east，
如果不能确认，余下的行程将被放弃。

STATUS

Syntax: #ST text

定义状态条，text 中可以显示变量，每当变量改变，状态条也随之改变。

实例：

#st {真气: @hp 内力: @nl 潜能:@qn 道行: @dx}

在状态条中显示有关数据

STEP

Syntax: #STE
Related: #SLOW #STOP #OK

恢复被放弃的沿路径慢行

STOP

Syntax: #STO
Related: #SLOW #STEP #OK

放弃继续沿路径慢行，通常在触发中使用。

实例：

#TRIGGER {五庄观第三代弟子 咖啡(%w)} {#STOP;kill %1}

沿路杀人越货，很刺激吧？

STW

Syntax: #STW string
Related: #STATUS

设置状态窗口，状态窗口类似与状态行，但能包括更多内容，

设置%ansi颜色并用%cr换行。可以使用菜单命令window/status
打开，在其上单击右键编辑。

实例：

#stw {当前道行：@dx %cr 初始道行：@dx0 %cr 战斗时间：@time1}

显示获取道行的速度

T+

Syntax: #T+ classname

激活触发类

T-

Syntax: #T- classname

关闭触发类，适当的运用以上两个命令可以避免机器人误动！

T?

Syntax: #T?
Related: #TIMER #TS

显示计时器（timer）的剩余时间

TAB

Syntax: #TA word

增加一个词到<TAB>键扩展列表

TELEPORT

Syntax: #TE room [zone]
Related: #WALK

改变你在地图上的位置，相应的MUD世界中的位置不变。
room可以是short name或room number。

TIMER

Syntax: #TI
Related: #T? #TS

开关计时器，对剩余时间没有影响。

TRIGGER

Syntax: #TR pattern command [classname]

建立或显示一个触发，与#action功能相同

TS

Syntax: #TS [value]
Related: #TIMER #T?

设置计时器时间或重新计时，value 指明以秒为单位的时间间隔，
如果省略，则重新计时。

TYPE

Syntax: #TY filenum [pattern]
Related: #FILE

在屏幕上显示文本文件，如果包括pattern，则只显示包含pattern
的行。pattern中可以包括通配符。

TZ

Syntax: #TZ
Related: #TS

将计时器置零

UNALIAS

Syntax: #UNA alias
Related: #ALIAS

删除一个别名

UNGAG

Syntax: #UNG
Related: #GAG

避免某行被忽略，通常用在触发中undo #gag命令。

实例：

#tr {咖啡告诉你} {#gag}

#tr {来吃人参果} {#ungag}

UNKEY

Syntax: #UNK key
Related: #KEY

删除一个宏键

UNTRIGGER

Syntax: #UNT pattern
Related: #TRIGGER

删除一个指定pattern的触发

UNVAR

Syntax: #UNV variable
Related: #VAR

删除一个变量

UNTIL

Syntax: #UN expression commands

执行命令直到表达式为真（true or non-zero）

实例：

#until (#hp>50) {quit}

URL

Syntax: #URL url

在浏览器中打开一个URL

实例：

#URL http://61.128.193.21/~~tlt

注意需要两个~！参见通配符。

VARIABLE

Syntax: #VA variable value

变量赋值，变量前不需要@。
也可以使用variable = value or variable := value.

VERSION

显示zMUD版本

VERBATIM

Syntax: #VERB [value]

开关分析模式，value用于指定分析模式，效果与使用菜单命令相同。

WAIT

Syntax: #WA [time]

暂停进一步的处理直到从主机收到新的信息，如果指定time，
则暂停指定的时间，time以毫秒为单位。

WALK

Syntax: #WAL room

快速到达地图上指定的标记处，room 是被事先定义的short name。

WHILE

Syntax: #WH expression commands

当表达式为真（true or non-zero）时执行命令

WINDOW

Syntax: #WIN name [string]

打开一个新窗口

实例：

#FORALL @eqlist {#WIN status %i}

打开窗口status并在其中显示装备列表。

WIZLIST

显示zMUD的作者

WRAP

Syntax: #WR [column]

设置文本回绕，如果指定column，则在指定的列处换行。

WRITE

Syntax: #WR n value [rec]
Related: #READ

写一个value到文件，如果n在1～5间，表示文本文件，value将被追加在文件尾部，rec被忽略；如果n在6~10间，则value写入记录rec，省略rec时，value加在文件最后。

#tr {离开游戏} {#wr 1 {%ctime}}

记录在MUD中浪费的时间！

YESNO

#YE question yes-command no-command

显示一个带按钮的确认对话框，<ESC> 放弃执行命令。可以使用{按钮提示:command}格式，*号用于表示焦点所在，enter 立即执行。

实例：

#YESNO Where to you want to go today?
{Temple:.temple} {Guild:.guild}
{*Microsoft:#URL http://www.microsoft.com}

下面是Pattern中可以使用的特殊字符

* 匹配任何数量的字符或空格
? 匹配一个字符
%d 匹配任何数量的数字（0－9）
%w 匹配任何数量的字母（a-z）
%a 匹配任何数量的字母或数字（0－9，a-z）
%s 匹配任何数量的空格（spaces, tabs)
%x 匹配任何数量的非空格
[range] 匹配任何数量的在[range]中列出的字符
^ 强制从一行的开始进行匹配
$ 强制匹配到一行的结束
(pattern) 保存匹配的式样到参数％1～％9
~ 包括其中的字符不会被解释为特殊字符
{val1|val2|val3|...} 匹配其中列出的任何特殊的串
{^string} 不匹配其中包括的串

使用[range]时，你可以在其中列出需要的字符例如
[abc] 或使用范围 [a-c] 。

为了匹配特殊字符本身，可以使用~将特殊字符括住，
例如：~[test~] 将匹配字符串
[test] 而不是作为 [range] 来匹配。

只使用$可以匹配一个空行。

别名中使用参数的例子

我们在解谜过程中向npc提出的问题通常是以：

ask npc about here
ask npc about rumors
ask npc about name

这样三个命令开始的，把他们定义成别名可以节省一些时间，像这样：

#al askn {ask %1 about here;ask %1 about here;ask %1 about name}

可如果npc的名字有两个单词，或一堆同名的npc中你要问其中第二个，
就需要把别名中的参数改为%-1，此时执行askn bing 2或askn tian bing

就不会出错了。

为了更好的理解参数的作用，试试执行下面的命令：

#al tt test1 %1 test2 %2 test3 %-1 test4 %-2

看看执行{tt a1 a2 a3 a4 a5}的结果，一切都清楚了吧？

表达式
当执行运算时，如果所有参数均为数字，则执行数学运算，否则执行字
符串操作。以下是一些常用的表达式（v1和v2代表变量或另外的表达式）：

v1+v2 如果v1或v2不是数字，则执行字符串相加
v1-v2 从v1中减去v2
v1*v2 相乘
v1/v2 v1除v2，结果不含小数
v1\v2 取模
v1&v2 逻辑运算and
v1 and v2 同上
v1 | v2 逻辑运算or
v1 or v2 同上
v1 xor v2 逻辑运算xor
v1 = v2 如果v1等于v2返回真
v1 > v2 如果v1大于v2返回真
v1 < v2 如果v1小于v2返回真
v1 >= v2 如果v1大于等于v2返回真
v1 <= v2 如果v1小于等于v2返回真
v1 <> v2 如果v1不等于v2返回真
v1 != v2 同上
v1 =~ v2 如果表达式v1中的pattern包含v2，返回真
v1 ~= v2 同上
-v1 返回v1的负值
!v1 逻辑运算非

zMUD中的函数和变量

zMUD中的函数和变量

变量与别名非常相似。不同之处在于别名只能在一条命令的开始被扩展，而变量可以在任?

#VAR container waterskin

这条命令将字符串waterskin赋值给变量container。这样，fill @container命令将被扩展

如果在General Preferences设置中的Expand Vars可用，变量也可以直接在命令行上被扩?

fill <@container>

这条命令将被强制解释成fill waterskin。

系统变量
zMUD提供了一些预先定义的系统变量，与自定义变量不同的是，这些系统变量以字符%开头
下面是系统变量的简单介绍：

%action 最后一次触发所执行的命令。
%char 你在这个MUD中的ID。
%cr 换行。
%ctime 以秒为单位表示的你的连线时间。
%def 当前使用的特殊字符。
%host 当前连线的MUD的域名或IP地址。
%i 与%repeatnum相同。
%lastcom 最后被执行的命令。
%lastcom2 倒数第二个被执行的命令。
%lastcom3 倒数第三个被执行的命令。
%lastinput 我没试出来，谁能告诉我？:-(
%line 从MUD中得到的最后一行文本。
%line2 从MUD中得到的倒数第二行文本。
%line3 从MUD中得到的倒数第三行文本。
%param1 从最后一次触发中获得的第一个参数。也可写成%1。
%param2..%param99 从最后一次触发获得的其他参数。
%port 当前连线的端口号。
%random 产生一个0至99之间的随机数。
%repeatnum 当前循环命令的索引。可写成%i。（参见loop命令）
%selected 当前选中的文本或命令。
%selline 当前选中的行。
%selword 当前选中的单词。
%title 当前MUD的标题。
%trigger 引起最近一次触发的行。

%window 当前窗口的标题。

---

zMUD中的函数
下面是zMUD中定义的函数

%abs(i) 返回i的绝对值
%additem(s,list) 增加字符串s到字符串列表list中
%alias(s) 显示别名s的内容
%ansi(fore,back) 返回给定颜色的ANSI代码
%begins(s1,s2) 如果字符串s1在字符串s2的开头，返回true
%btncol(button,back,fore) 改变按钮的颜色
%btnimage(button,filename) 改变按钮的图案
%case(i,s1,s2,s3...) 如果i=1，返回s1，...，最多不超过8个
%char(i) 返回ASCII码所代表的字符，可用于显示系统定义的特殊字符
%color(fore,back) 返回颜色的属性
%concat(s1,s2,s3..) 字符串相加，最多不超过9个
%copy(s,i,n) 返回字符串s的一部分，从第i个字符开始，共n个字符
%ddeopen(serv,topic) 打开一个DDE连接
%ddeclose 关闭一个DDE连接
%dde(serv,topic,item) 使用DDE读取数据
%ddemacro(serv,topic,s) 使用DDE执行命令
%ddepoke(serv,topic,item,value) 使用DDE写数据（参见#DDE命令）
%delete(s,i,n) 删除字符串s中从第i字符开始的n个字符
%delitem(s,list) 从字符串列表list中删除字符串s
%ends(s1,s2) 如果s1在s2的尾部，返回true
%exec(s1,s2,...) 执行s1,s2,...
%expand(s) 扩展字符串s中的变量和函数
%eval(p) 返回表达式p的结果
%format(f,a,b,c,d...) 格式化值a,b,c,...，f的形式为%w.dx，其中w表示长度，d表示小位，x表示数据类型：s是字串，n是数字，f是浮点数，m是货币表示
%getglobal(name) 返回全局变量name的值，全局变量保存在.ini文件中
%grep(i,s) 搜索文件号i并返回与字串s匹配的行号
%if(expression, true-value,false-value) 如果表达式为true，返回true-value，否则?豬alse-value
%insert(p,s,i) 在字串s的第i个字符处插入字串p
%ismember(s,list) 如果字串s在字串列表list中，返回true
%isnumber(s) 如果字串s是数字，返回true
%left(s,n) 从字串s中截取最左边的n个字符
%leftback(s,n) 从倒数第n个字符开始，截取字串s最左边的部分

%len(s) 返回字串s的长度
%lower(s) 将字串s转成小写
%max(a,b,c,d...) 取最大值
%min(a,b,c,d...) 取最小值
%mod(a,b) 取模
%null(s) 如果字串s为空，返回true
%numwords(s,d) 返回字串s中的单词数，d为单词分隔符，默认值是空格
%number(s) 把字串s转换成数字
%pick(s1,s2,s3,...) 显示picklist，并让用户选择一个或一个以上的字串，如果选择的?
桓觯 祷氐闹狄詈分隔。参见#pick命令
%pos(p,s) 返回字串p在字串s中的位置，如果p不在s中，返回0/false
%proper(s) 将单词中除第一个字母外的其他所以字母转成小写
%prompt(v,p) 提示用户给变量v赋值，p表示口令模式，输入将不会回显
%random(i,j) 返回一个>=i并<=j的随机数，如果省略j，则返回0～i间的随机数
%read(i,rec) 从文件i中读取记录rec
%remove(p,s) 在字串s中删除子串p
%repeat(s,n) 重复返回字串s共n次
%replace(s,p,r) 在字串s中搜索字串p并替换成字串r
%right(s,n) 从字串s中截取最右边的n个字符
%rightback(s,n) 从倒数第n个字符开始，截取字串s最右边的部分
%setglobal(name,value) 给全局变量name赋值
%time(format) 返回当前的日期时间，如果format省略，将返回详细的日期时间信息，可?
趁ormat串中使用dd, mm, mmm, yy, hh, mm, ss,等返回所需要的信息
%trigger(class) 如果指定的trigger class 可用，返回true
%trim(s) 消除字串两端的空格
%trimleft(s) 消除字串左端的空格
%trimright(s) 消除字串右端的空格
%upper(s) 转换成大写
%word(s,i,d) 返回字串s中的第i个单词，d指出字串的分隔符，如果省略，默认是空格
%write(i,s,rec) 写字串s到文件i的第rec个记录。如果rec为0，字串写入文件尾部。对于
本文件，rec将被忽略，s将追加的文件中。
%yesno(s) 显示问题s和按钮yes/no，根据按下的按钮返回true/false。参见#yesno命令。
果显示的按钮超过两个，返回的将是按钮号，可与#case连用。如果只显示两个按钮，可与
f命令连用以充分发挥其用途

地图函数
地图函数用来存取地图数据。[]中是可选项。如果省略房间（room）或区域（zone）参数?
取当前房间或区域的值。房间号可以用代号（short name）代替。

%roomname(room, [s]) 返回房间名或用字串s命名房间

%roomdesc(room, [s]) 返回房间描述或用字串s描述房间
%roomnum(room) 返回一个房间的编号
%roomid(room, [s]) 返回房间的代号，或用字串s增加或修改房间代号（short name）
%roomcom(room, [s]) 返回或设置进入房间后的执行的命令
%roomnote(room, [s]) 返回或设置房间备注
%roomexit(room, [s]) 返回或设置房间出口字串。出口字串用|分隔。
%roomobj(room,) 返回或设置房间中对象的数量
%roommob(room,) 返回或设置房间中npc的数量
%roomcost(room,) 返回或设置进入房间所需的代价
%roomkind(room,) 返回或设置房间类型。0＝普通，1＝水，2＝飞，3＝陷井。加128于设置禁止入内标志
%roomflag(room,) 返回或设置禁止入内标志，0＝flase，1＝true
%roomlink(room, dir,) 返回或设置周围房间的编号。i=-1用于删除与某方向房间的接。i=-2表示未知的连接
%roomportal(room, s, , [z]) 返回或设置一个非标准的出口（portal）。i表示将到的房间号，z表示将到达的区域，s是判断是否通过非标出口的字串。参见地图详解
%numrooms() 返回当前区域的房间数目
%numzones() 返回当前地图中的区域数目
%parsemode(I) 返回或设置当前地图的分析模式。I=0表示详细分析，i=1表示简要分?璇=2分析look命令
%walk(i) 返回走到房间i的speedwalk字串
%zonename(zone, [s]) 返回或设置区域名
%zonenum(zone) 返回区域编号