

EasyUO 中文指令說明書

中文編譯，整理： ComIdiot

原文出處： EasyUO team,

Wes32980

Codename Alexandria: EasyUO Documentation

原文網址：<http://www.easyuo.com>

<http://codename-alex.sourceforge.net/easyuo-docs/>

目錄

版權聲明	9
改版記錄	9
關於 EasyUO	9
<i>EasyUO 基礎認識</i>	<i>11</i>
程式畫面介紹	11
基本知識	12
基本操作知識	14
EasyUO 中的變數種類	14
程式撰寫注意事項	15
EasyUO 的程式流程控制指令	17
<i>範例小程序</i>	<i>18</i>
輸入訊息	18
移動滑鼠	18
打開角色的包包和屬性視窗	18
打開魔法書(如果有的話)	19
把主包包中，沒放在袋子中的布剪成繃帶	20
敵視牛羊，但是不攻擊它，自動用繃帶補血	20
練隱身海上 8x8	22
<i>FAQ [常見問題與回答]</i>	<i>24</i>
運算子說明	26
<i>Commands [指令集]</i>	<i>33</i>
<i>Variables [系統變數]</i>	<i>34</i>
Call (呼叫副程式)	36
Chooseskill (選擇技能)	37
Click (滑鼠指令)	38
Cmppix (比較畫點顏色)	40

Contpos (設定容器位置)	41
Deletejournal (刪除日誌)	42
Deletevar (清空變數)	43
Display (對話視窗)	44
Event drag (拖拉物件)	45
Event macro (執行 UO 內建巨集)	46
Event pathfind (UO 內建路徑尋找)	53
Event skilllock (設定技能鎖定)	54
Event sysmessage (顯示系統訊息)	55
Execute (執行外部程式)	56
Event Sleep (暫停 UO 執行)	57
Exit (結束目前的程式)	58
Else (If-Else command)	59
Finditem (尋找指定物件)	60
Getshopinfo (取得商店資訊)	65
Gosub (執行副程式)	66
Goto (無條件跳躍指令)	68
Halt (終止程式執行)	69
Hideitem (隱藏物件顯示)	70
If (If-else 條件判斷指令)	71
Ignoreitem (令 FINDITEM 忽略物件)	73
Initevents (啟動事件指令功能)	75
Key (送出鍵盤按鍵)	76
Linespercycle (設定執行速度)	77
Menu (顯示自訂選單)	78
Move (移動角色)	83
Msg (說話)	84
Nextcpo (設定下個容器位置)	85
Onhotkey (設定熱鍵)	86

Return (結束副程式)	88
Pause (暫停程式執行)	89
Playcd (放 CD 音樂)	90
Savepix (儲存畫面某一畫點資訊)	91
Setshopitem (設定要買的商品)	92
Sleep (暫停程式執行)	93
Scanjournal (掃瞄系統日誌)	94
Send (送出 HTTP 封包)	96
Set (設定變數)	99
Setuotitle (設定 UO 標題顯示)	100
Shutdown (電腦關機)	101
Stop (停止程式執行)	102
Str (字串操作指令)	103
Terminate (結束 EASYUO)	105
Target (等待目標選取游標)	106
Uoxl (多 UO 操作指令)	107
Wait (等待一定時間)	108
#AR (物理抗性)	109
#CHARDIR (角色方向)	110
#CHARGHOST (是否死亡)	111
#CHARID (角色 ID)	112
#CHARNAME (角色名字)	113
#CHARPOSX (角色 UO 地圖 X 座標)	114
#CHARPOSY (角色 UO 地圖 Y 座標)	115
#CHARPOSZ (角色 UO 地圖 Z 座標)	116
#CHARSTATUS 角色狀態	117
#CLIVER UO 版本	118
#CLINR (目前操作的 UO 編號)	119
#CLICNT (多少個 UO 執行中)	120

#CLILEFT (遊戲視窗 X 座標)	121
#CLITOP (遊戲視窗 Y 座標)	122
#CLIXRES (遊戲視窗寬度)	123
#CONTID (容器 ID)	124
#CONTKIND (容器種類)	125
#CONTPOX (容器 X 座標)	126
#CONTPOSY (容器 Y 座標)	127
#CONTSIZE (容器大小)	128
#CONTTYPE (容器類別)	129
#CURSORX (滑鼠 X 座標)	130
#CURSORY (滑鼠 Y 座標)	131
#DEX (敏捷度)	132
#DATE (日期)	133
#JCOLOR (日誌顏色)	134
#JOURNAL (日誌變數)	135
#LHANDID (左手持物 ID)	136
#FINDBAGID (找到的容器 ID)	137
#RHANDID (右手持物 ID)	138
#LLIFTEDID (最後拿起的物件 ID)	139
#LOBJECTID (最後用的物件)	140
#LOBJECTTYPE (最後用的物件類別)	141
#FINDCOL (找到的物件顏色)	142
#LSKILL (最後用的技能)	143
#LSPELL (最後用的法術)	144
#FINDDIST (找到的物件距離)	145
#FINDID (找到的物件 ID)	146
#FINDKIND (找到的物件位置)	147
#FINDREP (找到物件的名聲)	148
#FINDMOD (設定物件偏移值)	149

#DISPRES (最後按下的對話視窗按鈕)	151
#FINDCNT (找到物件的總數)	152
#FINDSTACK (找到物件是否堆疊)	153
#FINDTYPE (找到物件的種類)	154
#FINDX (找到物件的 X 座標)	155
#FINDY (找到物件的 Y 座標)	156
#FINDZ (找到物件的 Z 座標)	157
#GOLD (金錢數)	158
#HITS (HP)	159
#INT (智力)	160
#LTARGETX (最後目標 X 座標)	161
#LTARGETY (最後目標 Y 座標)	162
#LTARGETZ (最後目標 Z 座標)	163
#SENDHEADER (HTTP 檔頭設定)	164
#TRUE/#FALSE (真/偽常數值)	165
#FOLMAX	166
#CLYRES (遊戲視窗高度)	167
#MAXWEIGHT (最大負重)	168
#ENEMYID (敵人的 ID)	169
#ENEMYHITS (敵人 HP 值)	173
#MAXHITS (最大 HP 值)	174
#MAXSTATS (最大屬性總和)	175
#FOLLOWERS (目前用掉的寵物控制數)	176
#CURSKIND (游標種類)	177
#LTARGETID (最後目標 ID)	178
#LTARGETKIND (最後目標種類)	179
#MANA (目前 MP 值)	180
#NEXTCPOSX (下個容器的 X 座標)	181
#NEXTCPOSY (下個容器的 X 座標)	182

#PIXCOL (畫點顏色)	183
#RANDOM (亂數)	184
#SCNT (系統計數器)	185
#SCNT2 (系統計數器 2)	187
#SEX (性別)	188
#SHARD (SHARD 名稱)	189
#SHOPCNT (商店商品總數)	190
#SHOPCURPOS (商店商品目前位置)	191
#SHOPITEMID (商店商品 ID)	192
#SHOPITEMMAX (商店商品數量)	193
#SHOPITEMNAME (商店商品名稱)	194
#SHOPITEMPRICE (商品價格)	195
#SHOPITEMTYPE (商品種類)	196
#SKILL (技能值)	197
#SKILLCAP (技能上限)	198
#SKILLLOCK (技能鎖定狀態)	199
#STAMINA (目前精力值)	200
Sub (副程式宣告)	201
#STR (力量值)	202
#STRRES (字串運算結果)	203
#SYSMSG (系統訊息)	204
#SYSMSGCOL (系統訊息顏色)	205
#TARGCURS (是否是選定目標的游標)	206
#TIME (電腦時間)	207
#WEIGHT (目前負重)	208
#MENUBUTTON (選單按鈕)	209
#MENURES (按下的選單按鈕)	210
#LTARGETTILE (最後目標圖形種類)	211
#SMC (分號“;”)	212

#SPC (空白)	213
相關網路資源	214

版權聲明

EasyUO 中文指令說明書，是由 ComIdiot 所編譯，即本文章除了翻譯，參考原文著作外，還有加入個人的見解和心得，嚴禁在未經本人的同意下，放至在網站上供人下載使用。嚴禁以任何方式將本文件輸出到其它的媒體上。本人保留隨時取消供人使用本文件之權利。

本文的範例程式部份，完全由本人所撰寫而成。主要供各位同好拿來做為學習的材料，以及實驗用途，嚴禁直接拿來使用，如程式有任何問題，或是造成你個人的損失，本人不負任何責任。在某些情況下，範例程式有可能會讓你的角色變成灰名，或是更糟的情況。在使用前，請先確定角色所在的環境，是否適合測試這些指令。

如有任何問題，可以在巴哈 BBS 上發信給我(blackfox)互相討論，恕不幫忙撰寫程式。另外，會翻譯，不代表我是 [EasyUO](#) 通，有些東西我根本是連用都沒用過，故請別抱太大的期望，認為我有問必達。

翻譯難免會有字誤，更嚴重者會有語誤，請來信指正。

Mirror 站台：目前無。想 Mirror 前請先詢問，謝謝!!

以下為速查頁

[指令快速索引](#)

[系統變數快速索引](#)

改版記錄

2003/5/xx	1.00 第一次出版
2003/6/xx	1.10 加入範例程式
2003/6/13	1.11 加入 STR 和 #STRRES
2003/6/18	1.12 訂正文字，指令加上中文標題說明；更新網路連結。
2003/6/19	1.12 增加一個 8x8 練隱身的例子，以及更新 #FINDDIST 的說明，增加連結及增加程式撰寫注意事項。

關於 [EASYUO](#)

[EasyUO](#) 並不是一個由 [OSI](#) 所認可的程式，對 OSI 來說，EasyUO 並不是合法的。所有由 [OSI](#) 所認可的程式，都會放在 UOPro 網頁中，如 [UOA](#)，[UOAM](#) 等。也就是說，使用這個程式，可能會影響到你玩 UO 的權利，使用不當者，甚至會被永久停權，都是有可能的。

[EasyUO](#)，和 [UOA](#) 不同，[EasyUO](#) 是由存取 UO 在電腦中執行時，在記憶體中

的資料，及模擬鍵盤和滑鼠的功能，來達到它所想要做的各種功能，它並不會傷害你的電腦，也不是所謂的加速程式。[EasyUO](#)並不能加速 UO 的執行，它只是可以讓你在玩 UO 時更方便。

[EasyUO](#)並不能支援其他的遊戲，而且它只有支援 UO 2D 的程式，3D 並不支援。

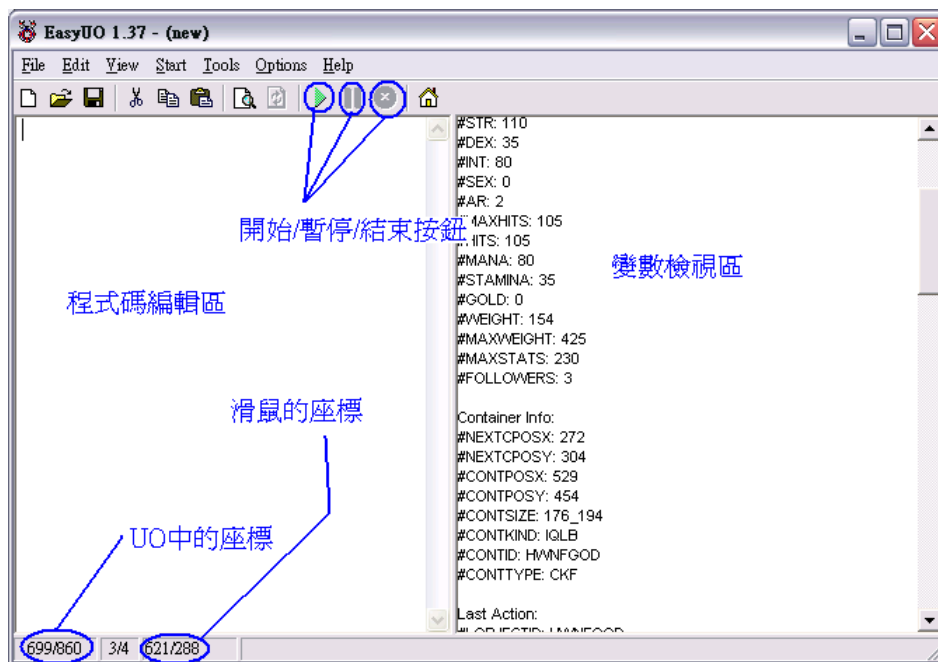
更多有關 EasyUO 的問題，請參考 EasyUO 官方網站的 [FAQ](#)

EasyUO 是一個已經寫得相當完善的直譯器，目前看到最大的程式可達 4000 行以上，是可以寫出很複雜動作的程式。

EASYUO 基礎認識

程式畫面介紹

首先，先到 <http://www.easyuo.com> 下載最近版的 EasyUO 後，解壓程式到一個目錄，然後執行 EasyUO.exe：



- 開始/暫停/結束按鈕：主要是用來操作程式的進行。
- 程式碼編輯區：在此編寫程式碼。
- 變數檢視區：看系統或是自訂的變數。
- 座標系統顯示：用來快速的看二種不同的座標現在的值。

「View」選單：用來設定要看的變數，和 EasyUO 的視窗狀態：

- Toolbar 選項：設定是否顯示工具列。
- StayOnTop：設定 EasyUO 視窗是否永遠在最前景。
- Variables：
 - Show：是否顯示變數檢視區視窗。
 - Standard：變數檢視區視窗顯示標準的系統變數。
 - User Define：變數檢視區顯示定義的自訂變數
 - Define..：設定當使用 User Define 時，要顯示哪些變數。

「Tools」選單：

- Design-Hotkeys：設定一些巨集，或是可說是程式片斷，可快速的把重覆的程

式片斷貼到程式編輯區中。

UOXL：EasyUO 內建的多 UO 功能，可以同時開啓多個 UO，理論上可開無限多個，直到記憶體爆掉…這個選項和 EasyUO 的一個指令：[UOXL](#) 是一樣的作用。

New Client：開一個新的 UO

Swap to next Client：切換到另一個已開啓的視窗。

「Options」：設定 EasyUO 的選項：

Pause Key：定義哪個按鍵是用來暫停的。

Configuration：設定 EasyUO 的一些開關：

Enable Event Sysmessage：可以使用 Event Sysmessage

Don't move cursor：EasyUO 在送出滑鼠的指令時，只送給 UO，螢幕上的游標不會受到影響。

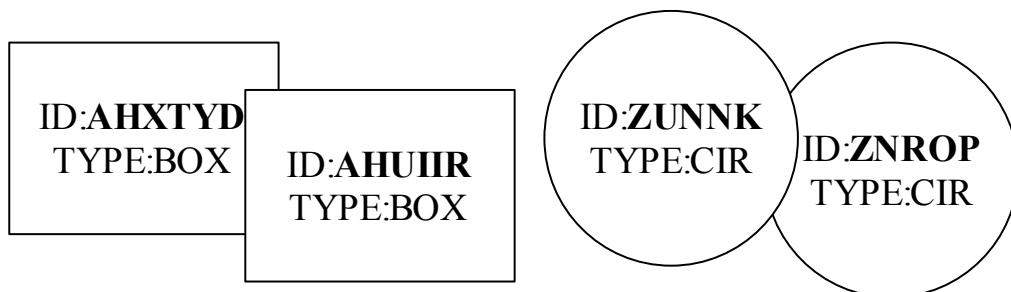
Allow execute：允許執行外部程式。

Allow send：允許 [SEND](#) 指令可用，可送出 http 封包到外部網頁主機。

Remember settings：基本上在 Configuration 中的設定，EasyUO 是不會記憶的，但是在選了這個選項後，設定會存起來，並在下次開啓時，會載入這些設定。

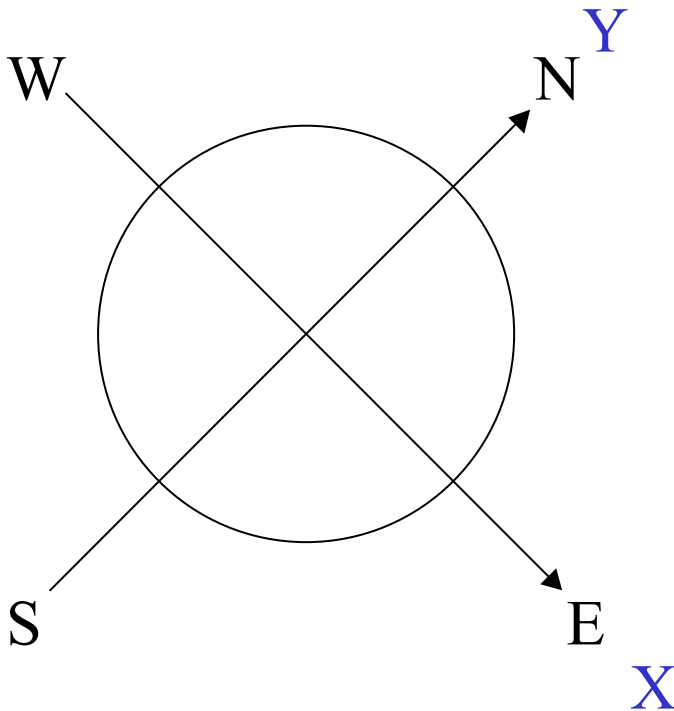
基本知識

- 物件 ID(Object ID)：在 UO 世界中，所有的東西都是有一個所謂的識別號碼的，就是所謂的 ID。OSI 利用物件 ID，可得知東西的去向，而技能會不會提昇，其實是和物件 ID 有關係的，例如馴獸技能，或是開鎖技能，主機都會記錄最近幾次(或可說是幾百次)用過技能的物件 ID，以避免我們重覆對同一對象使用同一種技能，來達到提昇技能的目的(技能不好練，A 錢才多??)。



- 物件種類(Object Type)：同一種類型的物件，它們的物件種類是一樣的，例如每一頭牛的 ID 不同，但是他們的種類都是相同的數值「IG」。生物在死

- 後會變成屍體，由上可知所有的屍體都會有同一個種類字串，它是「YFM」。
- 當命令 EasyUO 在找物件時，如果依 ID 來找而且有找到的話，一定只找到一個，但是如果以種類來找的話，可能會找到很多個。
- 最後用的物件(Last Object)指的是你最後點選(拿取)的東西，例如刀子，木板等等。
- 最後選的目標(Last Target)：指的是最後選的目標，例如點二下刀子，然後出現照準游標後，點一下屍體，此時最後用的物件是刀子，但是最後的目標是用刀子割的屍體。
- 地圖座標系統：在 UO 內部，座標其實不是六分儀系統，而是單純的 XY 二維座標，要得知人物的座標，只要看 EasyUO 左下角的顯示數字，或是察看 [#CHARPOSX](#) 和 [#CHARPOSY](#) 這二個系統變數就可以了。[#CHARPOSZ](#) 代表的是角色目前的高度。如下圖：



- 螢幕座標系統：主要是用來決定，得知游標的位置，可由[#CURSORX](#) 和 [#CURSORY](#) 來得知，要設定游標的位置，請用 [CLICK](#) 指令。
- 何時使用哪種座標系統：基本上，人物的移動時要用 UO 的座標，找到地面上的東西時，系統變數放的是 UO 的座標，而要移動包包，包包中的東西等他人其在 UO 程式中看不到的物件時，就要用螢幕座標。詳情請看 EasyUO 中很常用，也非常複雜的指令：[FINDITEM](#)。
- 最好在 UO 是英文版的模式下執行 EasyUO。如何設成英文模式，請參考 [FAQ](#)。

基本操作知識

這裡要介紹如何得到一些在很多的程式中都要設定的資訊。

在程式中，我們常可看到在程式一開始時，會要求設定什麼匕首的 ID 啦，剪刀的 ID 啦，在此我介紹三種方式，可以找到要的 ID：

- 刀，劍等不會用掉的物件(不能吞下去的)，直接在包包中點要看的物件二下，它們的 ID 會出現在[#OBJECTID](#) 中，種類會出現在[#OBJECTTYPE](#) 中。
- 對象是生物時，雙點它是沒有用的，這時，請點二下包包中的剪刀或是可以割東西的武器，在出現選取的游標時，點向要查的生物，則它的 ID 會出現在[#LTARGETID](#) 中。
- 由上一點我們沒有辦法得知生物的種類字串是什麼，也沒有系統變數可直接查，有一個看來很像的變數[#LTARGETKIND](#)，但是它的意義，並不是代表目標對象的種類。此時，可以寫一小段程式來看生物的種類：

```
FINDITEM #LTARGETID
HALT
```

執行完後，我們查一下[#FINDTYPE](#) 這個變數，其中存的就是該生物的種類值。記得最後要加上 HALT 指令，否則程式會一直重覆的重頭執行。

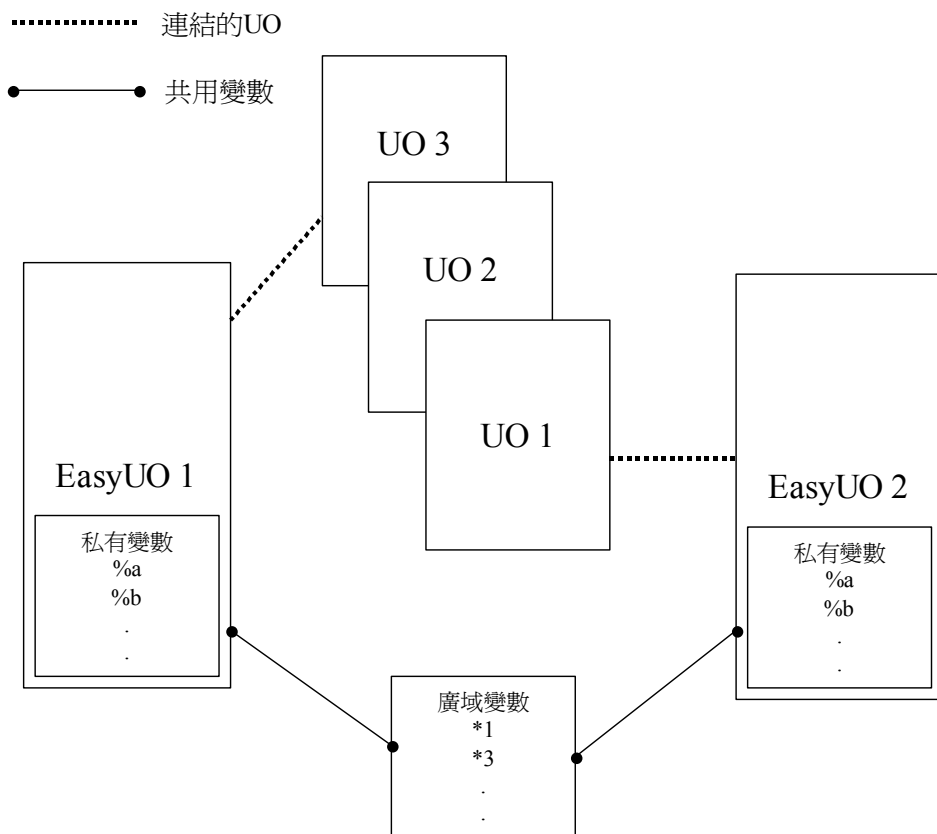
- 在 UO 中有 LastTarget, LastSkill 等等巨集，這些內建的巨集，全都可在 EasyUO 中使用。在使用這些功能前，相關的系統變數要先設好；例如使用 LastTarget 前，如果對象是生物，只要設[#LTARGETID](#) 就可以了，但是如果對象不是生物，例如挖礦，你要設定的是最後一個點的 X, Y, Z 三個座標值。這三個座標分別放在[#LTARGETX](#)，[#LTARGETY](#) 和[#LTARGETZ](#) 中。
- EasyUO 中有一些變數，例如現在的法力[#MANA](#) 等變數，需要先把屬性視窗打開，才可正確的顯示。

EASYUO 中的變數種類

在 EasyUO 中，其有三種類的變數可以使用：

- 以「#」為開頭的變數：這一種變數是所謂的系統變數，系統變數不多是和 UO 內部的一些變數相連結，它是否就是 UO 的變數本身的存放位址，則不得而知。例例如最後用到的物件，就可在[#OBJECTID](#) 這個變數中存著最後用的物件 ID。
- 以「%」為開頭的變數，是所謂的私有變數。私有變數只存在 EasyUO 這個程式本身。打個比方，如果你開啓了二個 EasyUO 而每一個 EasyUO 都有一個變數，名字叫%temp，那麼，在這二個 EasyUO 中，%temp 的變數是互相沒有關係的。

- 以「*」為開頭，後面是數字(通常都會用 1-99 的數字)的變數，例如「*1」，「*8」等，是所謂的廣域變數。廣域變數的作用是在各 EasyUO 的程式中，都可以看到同一個值，故這樣可以用來做程式間的溝通。其實這種變數它是存在系統的登錄(Registry)中的。這三種變數，請參考下圖：



很少有程式會用到雙 EasyUO 的，即使是練魔防的程式，最多也是只要用一個 EasyUO，開二個 UO 視窗，再使用 [UOXL](#) 指令來切換 UO 的視窗就可以了。在全部看完本書後，你是否可以寫出以下的程式？

- 開二個角色，其中一個角色會自動跟著你操作的角色移動，但是不和你的角色重疊在一起。
- 二個角色會互相幫對方解毒程補血。
- 被動的角色(就是不是你在操作的那一隻)會主動攻擊你正在打的敵人。

這個程式是一定可以做到的，在看完本書的同時，你可以順便想想看，要達成這個程式所要的功能，要什麼樣的指令和邏輯？

程式撰寫注意事項

- ✓ 不可輸出中文：如程式的 [MSG](#) 指令，只可輸出英文字，因為 EasyUO 的撰寫人是德國人，所以他們沒有考慮過雙字元集(DBCS)的問題。
- ✓ 運算式注意事項：
 - 運算式間一定要有空白：如「if(a=3)或 if(a=3)」皆是會有問題的，要寫成「if(a=3)」。另一個例子：「if !%error」是不合法的，寫成「if !%error」。
- ✓ EasyUO 在你指令寫錯時，是不會告訴你的：常看到的是變數的「%」符號，或是系統變數的「#」符號忘了加，程式還是一樣執行，只是結果不對。
- ✓ 冒號後面一定是註解，想在一些指令如 [MSG](#) 中什冒號持，要用 [#SMC](#) 取代。
- ✓ 程式視窗有大小的限制：常見於 95/98/me 的系統中，但是前題是你的程式已大到四千行以上，才有可能發生。
- ✓ 最好在英文的 UO 下執行：如此可保障你拿來的程式，不能執行的機會下降。
- ✓ 某些指令和 UOA 或是自訂的 UO 熱鍵會發生衝突：最常看到的是 [MOVE](#) 指令。

EASYUO 的程式流程控制指令

在此列出 EasyUO 的程式控制指令，方便速查：

判斷式：

[IF](#)

[ELSE](#)

[CMPPIX](#)

分支和迴圈：

[FOR](#)

[CALL](#)

[SUB](#)

[GOSUB](#)

[RETURN](#)

[GOTO](#)

其它控制：

[EXIT](#)

[HALT](#)

[STOP](#)

[PAUSE](#)

範例小程式

在試著執行這些範例程式前，建議先看過本書的其他部份，再回頭來看這些程式，相信會有更大的收穫。這些例子都沒有說明，就留給各位去了解。

輸入訊息

[initevents](#)

[msg](#) this is a public message

msg [#SMC](#) . [#SPC](#) . this is a wisper message

msg /this is a note

[event sysmessage](#) This is a system message

[halt](#)

移動滑鼠

[for](#) %a 1 10 4

[set](#) %rx [#RANDOM](#)

set %ry [#RANDOM](#) / 2

[click](#) %rx %ry n

[wait](#) 4

halt

打開角色的包包和屬性視窗

```
.;*****
```

```
initevents
```

```
event sysmessage opening backpack..
```

```
event macro 8 7
```

```
wait 30
```

```
contpos 100 400
```

```
wait 30
```

```
event sysmessage opening status bar..
event macro 8 2
contpos 20 40
halt
;*****
```

打開魔法書(如果有的話)

```
;請在 800x600 的遊戲視窗下執行
;*****
initevents
ignoreitem reset
event sysmessage searching spellbook..
finditem MPF
if #FINDKIND = 0
{
    set #LOBJECTID #FINDID
    ignoreitem #FINDID
    event macro 17 0
    wait 30
    contpos 197 176
    wait 30
    click 252 191 d
    wait 30
    click 537 192
    wait 30
    click 363 233
    wait 20
    click 275 233 r
}
halt
;*****
```

把主包包中，沒放在袋子中的布剪成繃帶

主包包中要有布和剪刀

(記得用一點點布來試就好!!)

```

;*****
initevents
ignoreitem reset
event macro 8 7
wait 30
finditem KAG ;自己做的剪刀 Type，商店的剪刀 Type 可能不同
if #FINDKIND = 0
{
    set %scro #FINDID
doband:
;商店買來的布
finditem BUI
if #FINDKIND = 0
{
    set #LOBJECTID %scro
    set #LTARGETID #FINDID
    event macro 17 0
    target
    event macro 22 0
    wait 20
    goto doband
}
}
halt
;*****
    
```

敵視牛羊，但是不攻擊它，自動用繃帶補血

請注意：本程式未經測試，但是程式的邏輯是正確的。

```

initevents
ignoreitem reset
event macro 8 7
    
```

```

wait 30
contpos 50 800
wait 30
event macro 8 2
contpos 20 40
set %bandage NA
finditem ZLF C
if #FINDKIND = 0
{
    set %bandage #FINDID
}
set %healgap 16
set %ctime 0
set %cow IG_NG
set %sheep PF_FG
set %animals %cow . %sheep
waitfor:
    ignoreitem reset 20
    call attackanimal
    wait 20
    if #HITS < ( #MAXHITS - 20 )
    {
        if ( #SCNT2 - %ctime ) > %healgap
        {
            if %bandage <> NA
            {
                set #LOBJECTID %bandage
                event macro 17 0
                wait 20
                event macro 23 0
                wait 20
                set %ctime #SCNT2
            }
        }
    }
}
goto waitforkey

;***** subroutine

```

```

sub attackanimal
finditem %animals G_7
if #FINDKIND = 1
{
    ignoreitem #FINDID
    if #FINDREP <> 1 && #FINDREP <> 2 && #FINDTYPE <> X
    {
        set #LTARGETID #FINDID
        event macro 27 0
        wait 10
        event macro 6 0
    }
}
return

```

練隱身海上 8x8

;練隱身海上 8x8 , By ComIdiot

;須用英文界面

```

initevents
msg #smc raise anchor$
loop:
set %oldx #charposx
set %oldy #charposy
event macro 13 21
wait 1s
scanjournal 1
set %journal1 #journal
scanjournal 2
set %journal %journal , #journal
deletejournal
if well in %journal
    set %success #true
else
    set %success #false
if %success
{

```

```
for %i 1 8
{
    msg #smc one forward$
    wait 1s
}
if ( %oldx = #charposx ) && ( %oldy = #charposy )
{
    msg #smc furl sail$
    msg #smc turn around$
    msg #smc forward$
    wait 4s
    msg #smc furl sail$
}
}
if ! %success
{
    wait 9s
}
wait 3s
goto loop
```

FAQ [常見問題與回答]

1. 如何更改 UO 為英文環境?

結束 UO，到 UO 安裝目錄，開啓 UO.CFG，找一行字爲

「UserLanguageCodeString」的設定(如果找不到的話，自行加入)，把數值改成 UserLanguageCodeString=ENU，重開 UO 就可以了。要回復成中文，請把這一行刪除，或是在最前面加上分號，表示是註解。

2. EasyUO 程式好複雜! 哪裡有程式可以讓我直接可參考的!?

請到 [EasyUO 的討論區](#)(這是英文網頁)或是參考程式庫(英文)：

<http://217.110.204.13/tforum2/viewtopic.php?TopicID=10081>

本文章幾乎在每個指令後，都會有小小的範例，也是個不錯的學習的機會。

3. 什麼叫區塊程式碼段?

即由{}所包起來的程式段，可把它們視爲一個區塊(block)。

4. 螢幕的座標定義為何?

螢幕最左上角爲(0,0)，右下角則視解析度而定，例如在 1024x768 模式下，最右下角的座標爲(1023,767)。

5. 我用 EasyUO，怎麼覺得有時當機率會提高?

這是有可能的，因爲 EasyUO 是直接存取 UO 的記憶體，存取別人的記憶體，多少會有風險在。另外一個原因就是程式沒有寫好，例如你已坐上甲蟲了，卻硬是要用 EasyUO 的程式來產生用滑鼠雙擊甲蟲的動作，這種平常不可能發生的動作，當然會提高當機率，一個寫的好的程式，是不太會造成 UO 當機的。

6. 程式停掉後，再按執行，怎麼好處怪怪的?

這可能是有一些變數已用過，但是重新執行時，變數沒有初始化所造成。最簡單的方法就是重開 EasyUO。

7. 我用 EasyUO 會不會被 ban??

你如果在螢幕前看著它在執行，就算有 GM，我想應也不是問題吧! 但是如果用它來做所謂的離機 macro(人不在螢幕前，讓 EasyUO 一直跑)，被抓到了，可不能怪任何人。根據 OSI 的新規定，只要被抓離現在的地點，就會被記點，真的要用 EasyUO 做離機 macro，請多多考慮。

8. 你能不能幫我寫一支程式，如 xxxx

對不起，打這個中文說明，我覺得應對你有很大的幫助了→如果你可以好好的看完這份近二百頁，二萬多字的說明，想要不會寫也很難。某些地方有問題，還是歡迎來信討論。

9. EasyUO 是加速器嗎?

EasyUO 沒有對封包做任何的修改，基本上，EasyUO 是讀取/修改記憶體的內

容，並送出滑鼠和鍵盤指令等動作，完成沒有所謂加速器的行為。目前合法的所謂 Packet-stream 的 UO 輔助程式，就是 UOAssist。

10.

運算子說明

運算子：

目前除了「.」和「,」二個運算子只可用在 [SET](#) 和 [IF](#) 指令外，其他的運算子都可以在任何地方使用。另外，在運算式的各個元素間，和區塊程式碼符號{}，一定要用空白分開來。

運算子優先順序：

如果一個運算式中有數個運算子的話，EasyO 基本上會用內定的運算子的先後順序，依順來做運算。順序高的會先計算，例如乘法就比加法的順序來得高，故如 $2 + 3 * 4$ 這個式子，結果會是 $2 + 12 = 14$ 。

順序(1→高)	運算子
1 (*1)	.
2 (*1)	,
3 (*2)	()
4	! ABS
5	* / %
6	+ -
7	< > <= >= <> IN NOTIN
8	&&
9	

(*1) 「.」和「,」這二個運算子高過任何其他的運算子，甚至比指令還高，例如：

dis . play yes , no ok → display yesno ok

h , a , l , t → halt

(*2) 除了「.」和「,」外，括號一定會優先運算。

範例：

```
set %r 5 + 3 * 2      ; 5 + ( 3 * 2 ) = 11
set %r 8 - 3 - 2      ; ( 8 - 3 ) - 2 = 3
set %r 1 + 2 * 2      ; 1 + ( 2 * 2 ) = (
set %r 1 + 2 * 2 * 4   ; 1 + ( ( 2 * 2 ) * 4 ) = 17
set %r ( 1 + 2 ) * 2 * 4 ; ( ( 1 + 2 ) * 2 ) * 4 = 24
```

布林運算子：

布林運算通常傳回的值就只有真和偽二種。在布林運算中，0 代表偽，1 或是不為 0 的數代表真，[#TRUE](#) 和 [#FALSE](#) 分別代表真和偽。以下的簡寫 T 代表真，F 代表偽。

範例：

```
IF #TRUE
msg This will be executed$
IF #FALSE
msg This will not be executed$
```

EasyUO 有以下的布林運算子

!(NOT)

反轉布林值，即真變偽，偽變真。

	T	F
NOT	F	T

範例：

```
IF ! #FALSE
msg This will be executed$
IF !! #TRUE
msg This will be executed$
```

&& (AND, 和)

全都為真時才為真。

AND	T	F
T	T	F
F	F	F

```
IF #FALSE && #FALSE
msg This will NOT be executed$
IF #FALSE && #TRUE
```

```

msg This will NOT be executed$
IF #TRUE && #FALSE
msg This will NOT be executed$
IF #TRUE && #TRUE
msg This will be executed$
IF !( !#FALSE && #TRUE )
msg This will NOT be executed$

```

|| (或, OR)

只要有其中一個為真即是真。

OR	T	F
T	T	T
F	T	F

```

IF #FALSE || #FALSE
msg This will NOT be executed$
IF #FALSE || #TRUE
msg This will be executed$
IF #TRUE || #FALSE
msg This will be executed$
IF #TRUE || #TRUE
msg This will be executed$
IF #FALSE || #TRUE && #FALSE || #FALSE && #TRUE || !#FALSE
msg This will be executed$
IF ( #FALSE || #TRUE ) && ( #FALSE || #FALSE ) && ( #TRUE || !#FALSE )
msg This will NOT be executed$

```

< > <= >= = <> (比較運算：小於，大於，小於等於，大於等於，等於，不等於)

```

IF 1 = 3 || -3 < -2 || 4 = 4 && 5 = 8
msg This will be executed$

```

IN NOTIN (搜尋運算子)

```
IF he IN hello ;hello 中是否有 he 這二個字
    msg This will be executed$
IF 6 NOTIN 123457890 ;123457890 中是否沒有 6 這個字
    msg This will be executed$
```

位元運算子：

正常的情況下，位元運算是很少會用到的，位元運算子是對二個數做位元的運算，並傳回值，可以用的位元運算子共三個：&&，||和!。這時不可用真偽值來看計算的結果。

例子：

```
set %andv 3 && 5 ;%andv = 1
```

下圖是把 3 和 5 變成二進位數，來做位元運算的結果：

3=	0	0	0	0	0	0	1	1
5=	0	0	0	0	1	0	0	1
	0	0	0	0	0	0	0	1

=1

又如以下：

```
set %orv 3 || 5
```

3=	0	0	0	0	0	0	1	1
5=	0	0	0	0	1	0	0	1
	0	0	0	0	1	0	1	1

=7

每個位元做 OR 運算，得到的結果是 7

又如以下的運算

```
set %cmpv 3 || 7 && 5
```

會變多少? 答案是 7 (7 && 5 = 5, 5 || 3 = 7)

位元運算在 EUO 中不太用得到，故如果還是不太了解的話，可以先不理會這個功能。

「.」陣列建構子

「.」可以用來在 EUO 中建構陣列。在 EUO 中，陣列其實就是一個由常數開始，變數結束的集合，以下這二個敘述是一樣的：

```
set %test . 1 hello
set %test1 hello
```

上二行唯一的差別是%test1 是一個固定的變數，而在另一行中，%test.1 中的 1 是可以改變的變數，如以下這一段程式碼，可以設定 100 個變數的初始值：

```
for %cnt 1 100
{
set %test.%cnt #random % 10 + 1
}
```

由於「.»是最先被處理的，故變數這 100 個變數的名稱為%test1~%test100，而且它們的初始值會被指定為 1~10 的數字

以下的運算式是無法成立的，因為「.»是最先運算，故會出現錯誤：

```
set %test. (%cnt * 2) 0 ; doesn't work!
```

改成如下：

```
set %tmp %cnt * 2
set %test.%tmp 0 ; works!
```

另外，目前也不支援多維陣列，就是在運算式中用二個「.»，例如：

```
set %a 1
set %b 2
set %matrix.%a.%.%b test
msg %matrix1_2 $
```

請注意「.»和「.,」這二個運算子的運作方式，在 EasyUO 中有可能會改變，未來也有可能會出現更好的運算法，取代這二個運算子。

以下這二行程式，會產生一名叫%test 的變數，並把它值設為 0。我們可以用這個方法把變數的值傳給副程式，而用直接去參考到變數本身的值。

```
set %varname test
set %.%varname 0
```

以下的程式可以解決多重陣列的問題：

```
creating multi dimensional arrays in a smart way:
```

```

for %z 1 2
{
  for %y 1 2
  {
    for %x 1 2
    {
      set %dummy %x + %y + %z
      gosub set myarray %i %j %z %dummy
    }
  }
}
for %z 1 2
{
  for %y 1 2
  {
    for %x 1 2
    {
      gosub get myarray %x %y %z
      msg myarray %i %j : %return
    }
  }
}
halt

```

```

sub get
set %varname %1
for %cnt 2 %0
{
  set %varname %varname , _ , %cnt
}
set %return % . %varname
return

```

```

sub set
set %varname %1
set %tmp %0 - 1
for %cnt 2 %tmp
{

```

```

    set %varname %varname , _ , %cnt
}
set % . %varname % . %0
return

```

再聲明一次，在這所討論到的，是非常進階的程式設計技巧，你不需具備這些能力，就可以開始撰寫 EasyUO 的程式。

「,」運算子

「,」逗號運算子主要是用來串接字串用的它會把二邊的字串連接起來，變成一個完整的字串：

```

set %a HELLO_
set %b THERE
msg %a , %b $           ;HELLO_THERE

```

如果真的要使用逗號在字串中，那麼你可用以下的方法：

```

set %skill 99 , ,
set %skill %skill , 9
msg My magery is at %skill $

```

還有二個特殊的字元，空白和分號這二種字元，要在字串中使用它們的話，請參考[#SPC](#)和[#SMC](#)的說明。

.

Commands [指令集]

CALL	CHOOSESkill	CLICK	CMPPIX	CONTPOS
DELETEJOURNAL	DELETEVAR	DISPLAY		
ELSE	EVENT DRAG	EVENT MACRO	EVENT PATHFIND	
	EVENT SKILLOCK	EVENT SLEEP	EVENT SYSMESSAGE	
EXECUTE	EXIT			
FINDITEM	FOR			
GETSHOPINFO	GOSUB	GOTO		
HALT	HIDEITEM			
IF	IGNOREITEM	INITEVENTS		
KEY				
LINESPERCYCLE				
MENU	MOVE	MSG		
NEXTCPOS				
ONHOTKEY				
PAUSE	PLAYCD			
RETURN				
SAVEPIX	SCANJOURNAL	SEND	SET	
	SETSHOPITEM	SETUOTITLE	SHUTDOWN	SLEEP
STOP	STRC	SUB		
TARGET	TERMINATE			
UOXL				
WAIT				

VARIABLES [系統變數]

R = 可讀；R/W = 可讀寫

#AR (R)	#FINDCOL (R)	#MAXWEIGHT (R)
#CHARDIR (R)	#FINDDIST (R)	#MENUBUTTON (R/W)
#CHARGHOST (R)	#FINDID (R)	#MENURES (R)
#CHARID (R)	#FINDKIND (R)	#NEXTCPOSX (R/W)
#CHARNAME (R)	#FINDMOD (R/W)	#NEXTCPOSY (R/W)
#CHARPOSX (R)	#FINDREP (R)	#PIXCOL (R)
#CHARPOSY (R)	#FINDSTACK (R)	#RANDOM (R)
#CHARPOSZ (R)	#FINDTYPE (R)	#RHANDID (R/W)
#CHARSTATUS (R)	#FINDX (R)	#SCNT (R/W)
#CLICNT (R)	#FINDY (R)	#SCNT2 (R/W)
#CLILEFT (R/W)	#FINDZ (R)	#SENDHEADER (R/W)
#CLINR (R)	#FOLLOWERS (R)	#SEX (R)
#CLITOP (R/W)	#FOLMAX (R)	#SHARD (R)
#CLIVER (R)	#GOLD (R)	#SHOPCNT (R)
#CLIXRES (R/W)	#HITS (R)	#SHOPCURPOS (R)
#CLYRES (R/W)	#INT (R)	#SHOPITEMID (R)
#CONTID (R)	#ICOLOR (R/W)	#SHOPITEMMAX (R)
#CONTKIND (R)	#JOURNAL (R)	#SHOPITEMNAME (R)
#CONTPOSX (R)	#LHANDID (R/W)	#SHOPITEMPRICE (R)
#CONTPOSY (R)	#LLIFTEDID (R)	#SHOPITEMTYPE (R)
#CONTSIZE (R)	#LOBJECTID (R/W)	#SKILL (R)
#CONTTYPE (R)	#LOBJECTYPE (R)	#SKILLCAP (R)
#CURSKIND (R)	#LSKILL (R/W)	#SKILLLOCK (R)
#CURSORX (R)	#LSPELL (R/W)	#SMC (R)
#CURSORY (R)	#LTARGETID (R/W)	#SPC (R)
#DEX (R)	#LTARGETKIND (R/W)	#STAMINA (R)
#DATE (R)	#LTARGETTILE (R/W)	#STR (R)
#DISPRES (R)	#LTARGETX (R/W)	#STRRES (R)
#ENEMYHITS (R)	#LTARGETY (R/W)	#SYSMSG (R)
#ENEMYID (R)	#LTARGETZ (R/W)	#SYSMSGCOL (R/W)
#FALSE (R)	#MANA (R)	#TARGCURS (R/W)
#FINDBAGID (R)	#MAXHITS (R)	#TIME (R)
#FINDCNT (R)	#MAXSTATS (R)	#TRUE (R)

[#WEIGHT](#) (R)

CALL (呼叫副程式)

COMMAND: CALL 檔名 [參數 1 參數 2 參數 3 ...]

說明：本指令是用來呼叫其它檔案的程式碼，在呼叫的程式結束後，程式會回到原來呼聲的程式中。內定的程式碼副檔名是.TXT。

對於複雜的呼叫來程式，建議把可重覆使用的部份，分成另一個程式檔，如此程式可便於重覆使用，且較易偵錯。

CALL 亦可傳送參數給被呼叫的程式檔。被呼叫的程式可由以下的變數接收參數：

%0: 傳回總共有幾個參數傳過來

%1, %2, %3 ... %1 表示第一個參數，%2 表示第二個，依此類推。

如果在 A 叫程式 B，而 B 再呼叫另一個程式 C 時，如 B 有傳參數 C 時，則 A 傳給 B 參數會被覆蓋掉。所以在呼叫其它程式前，記得要先把用到的變數存起來。

範例:

```
call standardsubs           ;內定副檔名.txt
call standardsubs.txt       ;和第一行結果一樣
call mine.euo 2452 3456 200
call msg.txt goodbye
halt
**msg.txt**                msg %1 $ ;%1 = goodbye
exit          *****end*****
```

其他參考: [EXIT](#), [GOTO](#), [GOSUB](#), [FOR](#)

CLICK (滑鼠指令)

COMMAND: CLICK X Y [R D G P F X N]

說明：使用 CLICK 指令來下達任何有關滑鼠的動作。格式為：

CLICK X Y [option]

其中：

X: 螢幕 X 座標位置

Y: 螢幕 Y 座標位置

Option: 設定行為模式：

空白/不加任何參數：單擊滑鼠

R：點滑鼠右鍵

D：雙擊

G：拖動物件

P：放下物件

F：快速點擊

X：快速連點

X n：快速點 n 下

N：只有移動滑鼠游標，不做任何動作

目前座標的位置可由[#CURSORX](#)和[#CURSORY](#)來得知，或是由 EasyUO 中左下角的座標值取得

使用拖拉的指令可以拖拉物件(CLICK X Y G 和 CLICK X Y P 搭配使用)，另一種拖拉物件的方式是使用 [EVENT DRAG](#) 指令，請參考 [EVENT DRAG](#) 和 [CONTPOS](#) 這二個指令來了解相關的動作。

使用這個指令，不管 EasyUO 和 UO 是否在前景，滑鼠都會移動。如果不想讓滑鼠移動的話，請把 EasyUO 的「Don't Move Cursor」選項打勾。在某些狀況下這個選項也許會沒有作用。

☞當 Don't move cursor 的選項打開時，雙擊的動作有時會失效。

範例:

click 200 300 ; Regular/Single

clickclick 200 300 r ; Right Click

- click 200 300 d ; Double Click
- click 200 300 g ; Drag Click
- click 200 300 p ; Drop Click
- click 200 300 f ; Fast Click
- click 200 300 x ; Multiple Fast Clicks
- click 200 300 x 10 ; 10 Multiple Fast Clicks (10 is the specified number of clicks to perform)
- click 200 300 n ; Nothing or Null Click (just moves the cursor)

其他參考: [#FINDMOD](#), [#CURSORX](#), [#CURSOR_Y](#), [#CURSKIND](#), [EVENT DRAG](#)

CMPPIX (比較畫點顏色)

COMMAND: CMPPIX 數字 [T F]

說明:本指令會比較由 [SAVEPIX](#) 指令所存下來的點的顏色和目前同一點的顏色是否相同，假如相同的話，本指令的下一行或是下一個程式區塊就會執行。如果加上 f 這個參數的話，則是比較顏色不同時會執行。

範例 1:

```
cmppix 1
cmppix 1 t
cmppix 1 t 5
cmppix 1 f 5
```

範例 2:

```
savepix 570 30 1
N1:
cmppix 1 f
{
    halt
}
goto N1
```

範例 2 會把(570,30)之個點的顏色存在顏色的記憶區#1，然後會一直比較現在的顏色和存起來的顏色是否不同，如果不同的話，就會停止程式的執行。要試驗這個程式，請重開 UO，並在登入畫面時開始執行這個程式，並把滑鼠移到結束(Quit)的按鈕上，程式就會馬上結束。

其他參考: [SAVEPIX](#)

CONTPOS (設定容器位置)

COMMAND: CONTPOS X Y

說明：把最後一個打開的容器移到指定的 X,Y 位置，設定的 X,Y 座標指的是容器的左上角的座標值。在這裡所謂的容器包括一般的包包，袋子，或是工匠選單，屬性視窗等等。

範例 1:

```
msg bank$           ;銀行，給我錢!!  
wait 40  
contpos 10 10       ;移動銀行的保險箱到 10,10  
halt
```

範例 2:

```
INITEVENTS  
event macro 8 7     ;打開角色的包包  
wait 50             ;等一秒  
contpos 500 400     ;移動包包到 500,400  
halt
```

其他參考: [#CONTPOSX](#), [#CONTPOSY](#), [#CONTKIND](#), [#CONTTYPE](#), [#CONTID](#),
[#CONTSIZE](#)

DELETEJOURNAL (刪除日誌)

COMMAND: DELETEJOURNAL

說明：使 [SCANJOURNAL](#) 忽略現在的最後一行 Journal 訊息及之前的訊息(即目前所有日誌的內容都忽略)。主要是在使用 [SCANJOURNAL](#) 時，不會重覆掃到舊的訊息。

範例:

;假如有人說 hail 的話，你會回答 farewell 一次

```
for %cnt 10 1
```

```
{
  scanjournal %cnt
  if hail in #journal
  {
    msg farewell!$
    deletejournal
  }
}
```

其他參考:[SCNAJOURNAL](#)

DELETEVAR (清空變數)

COMMAND : DELETEVAR 變數名

說明：把指定變數內容清空。

範例：

```
Deletevar %posit1
```

其他參考：[SET](#)

DISPLAY (對話視窗)

COMMAND: DISPLAY 種類 訊息

說明：開出一個對話視窗，產生幾類的按鈕，並顯示設定的文字，文字只限定是英文文字。DISPLAY 可以產生的按鈕種類的參數為：

「OK」：ok

「OK」 「Cancel」：okcancel

「Yes」 「No」：yesno

「Yes」 「No」 「Cancel」：yesnocancel

哪個按鈕被按下，值會存在#DISPRES 中。

範例:

```
display yesno You are overloaded!$ Continue anyway?
if #dispres = yes
    msg /Answer was yes!$
if #dispres = no
    msg /Answer was no!$
halt
```

其他參考:[#DISPRES](#)

EVENT DRAG (拖拉物件)

COMMAND: EVENT DRAG 物件 ID

說明：這個指令可用來執行拉動物件的動作，而且不會動到滑鼠。並且不管動到的物件，是在容器中還是在地上，使用這個指令，不必設定#FINDMOD 這個變數。但是要放下物件時，要用 [CLICK](#) X Y P 這個指令來放下東西。

請注意，EVENT DRAG 後面的參數是物件的 ID，而不是物件的位置。另外這個物件如果不是在地上或是打開的容器中，UO 程式很有可能會當掉。

☞使用這個指令前，確定 [INITEVENTS](#) 指令，已先在這個程式檔中有執行過。

範例:

initevents

event drag #objectid ;拉動物件

msg \$;按下 Enter 鍵，對可堆疊的東西而言，就是拿全部。

click 123 456 p ;放到螢幕座標 123 456 的位置

其他參考:[CLICK](#), [INITEVENTS](#)

EVENT MACRO (執行 UO 內建巨集)

COMMAND: EVENT MACRO 參數 1 [參數 2] [參數 3]

說明：使用這個指令直接呼叫 UO 定建的巨集。要使用這個指令請先確定 [INITENEVTS](#) 已在這個程式檔中，至少執行過一次。

(感謝 WZA 更新 EVENT MACRO 的參數資料)

範例:

INITEVENTS

...

EVENT MACRO 下以下的參數(參數 3 須自行設定，如設 Hello 等)：

參數 1	參數 2	參數 3	說明
0	^		say
2	0	^	emote
3	0	^	whisper
4	0	^	yell
5	0		walk North West
5	1		walk North
5	2		walk North East
5	3		walk East
5	4		walk South East
5	5		walk South
5	6		walk South West
5	7		walk west
6	0		toggle War/Peace
7	0		paste
8	0		open Configuration
8	1		open Paperdoll
8	2		open Status
8	3		open Journal
8	4		open skills
8	5		open spellbook
8	6		open Chat
8	7		open Backpack

8	8	open Overview
8	9	open Mail
8	10	open Party Manifest
8	11	open Party Chat
9	0	close Configuration
9	1	close Paperdoll
9	2	close Status
9	3	close Journal
9	4	close Skills
9	5	close Spellbook
9	6	close Chat
9	7	close Backpack
9	8	close Overview
9	9	close Mail
9	10	close Party Manifest
9	11	close Party Chat
10	0	minimize Configuration
10	1	minimize Paperdoll
10	2	minimize Status
10	3	minimize Journal
10	4	minimize Skills
10	5	minimize Spellbook
10	6	minimize Chat
10	7	minimize Backpack
10	8	minimize Overview
10	9	minimize Mail
10	10	minimize Party Manifest
10	11	minimize Party Chat
11	0	maximize Configuration
11	1	maximize Paperdoll
11	2	maximize Status
11	3	maximize Journal
11	4	maximize Skills
11	5	maximize Spellbook
11	6	maximize Chat
11	7	maximize Backpack

11	8	maximize Overview
11	9	maximize Mail
11	10	maximize Party Manifest
11	11	maximize Party Chat
12	0	opendoor
13	1	use skill Anatomy
13	2	use skill Animal Lore
13	35	use skill Animal Taming
13	4	use skill Arms Lore
13	6	use skill Begging
13	12	use skill Cartography
13	14	use skill Detecting Hidden
13	15	use skill Discordance
13	16	use skill Evaluating Intelligence
13	19	use skill Forensic Evaluation
13	21	use skill Hiding
13	23	use skill Inscription
13	3	use skill Item Identification
13	46	use skill Meditation
13	9	use skill Peacemaking
13	30	use skill Poisoning
13	22	use skill Provocation
13	48	use skill Remove Trap
13	32	use skill Spirit Speak
13	33	use skill Stealing
13	47	use skill Stealth
13	36	use skill Taste Identification
13	38	use skill Tracking
14	0	last skill
15	0	cast spell Clumsy
15	1	cast spell Create Food
15	2	cast spell Feeblemind
15	3	cast spell Heal
15	4	cast spell Magic Arrow
15	5	cast spell Night Sight

15	6	cast spell Reactive Armor
15	7	cast spell Weaken
15	8	cast spell Agility
15	9	cast spell Cunning
15	10	cast spell Cure
15	11	cast spell Harm
15	12	cast spell Magic Trap
15	13	cast spell Magic Untrap
15	14	cast spell Protection
15	15	cast spell Strength
15	16	cast spell Bless
15	17	cast spell Fireball
15	18	cast spell Magic Lock
15	19	cast spell Poison
15	20	cast spell Telekinesis
15	21	cast spell Teleport
15	22	cast spell Unlock
15	23	cast spell wall of Stone
15	24	cast spell Arch Cure
15	25	cast spell Arch Protection
15	26	cast spell Curse
15	27	cast spell Fire Field
15	28	cast spell Greater Heal
15	29	cast spell Lightning
15	30	cast spell Mana Drain
15	31	cast spell Recall
15	32	cast spell Blade spirits
15	33	cast spell Dispel Field
15	34	cast spell Incognito
15	35	cast spell Magic Reflection
15	36	cast spell Mind Blast
15	37	cast spell Paralyze
15	38	cast spell Poison Field
15	39	cast spell Summon Creature
15	40	cast spell Dispel Field
15	41	cast spell Energy Bolt
15	42	cast spell Explosion
15	43	cast spell Invisibility

15	44	cast spell Mark
15	45	cast spell Mass Curse
15	46	cast spell Paralyze Field
15	47	cast spell Reveal
15	48	cast spell Chain Lightning
15	49	cast spell Energy Field
15	50	cast spell Flame Strike
15	51	cast spell Gate Travel
15	52	cast spell Mana Vampire
15	53	cast spell Mass Dispel
15	54	cast spell Meteor Swarm
15	55	cast spell Polymorph
15	56	cast spell Earthquake
15	57	cast spell Energy Vortex
15	58	cast spell Resurrection
15	59	cast spell Air Elemental
15	60	cast spell Summon Daemon
15	61	cast spell Earth Elemental
15	62	cast spell Fire Elemental
15	63	cast spell Water Elemental
15	101	cast spell [N] Animate Dead
15	102	cast spell [N] Blood Oath
15	103	cast spell [N] Corpse Skin
15	104	cast spell [N] Curse Weapon
15	105	cast spell [N] Evil Omen
15	106	cast spell [N] Horrific Beast
15	107	cast spell [N] Lich Form
15	108	cast spell [N] Mind Rot
15	109	cast spell [N] Pain Spike
15	110	cast spell [N] Poison Strike
15	111	cast spell [N] Strangle
15	112	cast spell [N] Summon Familiar
15	113	cast spell [N] Vampiric Embrace
15	114	cast spell [N] Vengeful Spirit
15	115	cast spell [N] wither
15	116	cast spell [N] wraith Form

15	201		cast spell [C] Cleanse by Fire
15	202		cast spell [C] Close wounds
15	203		cast spell [C] Consecrate Weapon
15	204		cast spell [C] Dispel Evil
15	205		cast spell [C] Divine Fury
15	206		cast spell [C] Enemy of One
15	207		cast spell [C] Holy Light
15	208		cast spell [C] Noble Sacrifice
15	209		cast spell [C] Remove Curse
15	210		cast spell [C] Sacred Journey
16	0		last spell
17	0		last object
18	0		bow
19	0		salute
20	0		quit game
21	0		all names
22	0		last target
23	0		target self
24	1		arm/disarm Left
24	2		arm/disarm Right
25	0		wait for target
26	0		target next
27	0		attack last
28	0	^	delay
29	0		circletrans
31	0		close gumps
32	0		always run
33	0		save desktop
34	0		kill gump open
35	0		primary ability
36	0		secondary ability
37	0	^	set update range
38	0	^	modify update range
39	0		increase update range
40	0		decrease update range
41	0		maximum update range
42	0		minimum update range

43	0	default update range
44	0	update update range
45	0	enable update range color
46	0	disable update range color
47	0	toggle update range color

其他參考:[INTEVENTS](#)

EVENT PATHFIND (UO 內建路徑尋找)

COMMAND : EVENT PATHFIND X 座標 Y 座標 Z 座標

說明：這個指令可以讓你用 pathfinding(尋找路徑)的功能，座標表示的是目的位置的 UO 地圖座標，和 [MOVE](#) 不同的是，[MOVE](#) 指令是試著直線走到定點，而本指令是試著用 UO 內建的功能，也就是說它會「繞路」到目標點。Z 座標如果沒有給的話，內定是-1。

範例：

```
initevents
```

```
move 5010 4997
```

```
event pathfind 5020 4990 ;Pathfinding!
```

其他參考：[MOVE](#)

EVENT SKILLLOCK (設定技能鎖定)

COMMAND: EVENT SKILLLOCK 技能名 [UP DOWN LOCKED]

說明：設定技能的鎖定狀態為向上(UP)，向下(DOWN)或是鎖定(LOCKED)。要使用這個指令請先確定 [INTENEVTS](#) 已在這個程式檔中，至少執行過一次。

EVENT SKILLLOCK 的指令後面的技能參數，只要設定該技能的前四個英文字就可以，例如設定魔法技能向上的話，只要下達「EVENT SKILLLOCK MAGE UP」就可

唯一的例外是以下這二個技能：

Animal Lore (動物學) = ANIL

Stealth (潛行) = STLT

範例:

event skilllock mage down ;魔法技能向下
 event skilllock arms locked ;武器知識技能鎖定
 event skilllock necro up ;死靈技能向上

其他參考: [CHOOSESKILL](#), [#SKILL](#), [#SKILLLOCK](#)

EVENT SYSMESSAGE (顯示系統訊息)

COMMAND: EVENT SYSMESSAGE 訊息文字

說明：顯示一個系統訊息。系統訊息只存在 UO 程式中，別人是看不到的。要變更系統訊息的顏色，請使用[#SYSMSGCOL](#)。要使用這個指令請先確定[INITENEVTS](#)已在這個程式檔中，至少執行過一次。另外，要確定在 EasyUO 中，選項「Enable Event System Message」有打勾(內定是有打勾)

範例:

```
initevents  
event sysmessage This is a System Message.  
halt
```

其他參考: [MSG](#), [#SYSMSGCOL](#), [#SYSMSG](#), [SCANJOURNAL](#)

EXECUTE (執行外部程式)

COMMAND: EXECUTE 程式檔名及路徑 [參數...]

說明:使用 EXECUTE 來執行外部程式,並可傳參數給外部程式。請確定在 EasyUO 中,選項「Allow Execute」有打勾(內定是沒有打勾)。外部程式是以是執行檔,批次檔,或是電腦可開啓的任何文件,網址等。

範例 1:

```
execute easyuo.exe anotherscript.txt
```

```
execute http://www.easyuo.com
```

```
execute mailto:cheffe@easyuo.de
```

```
execute command.com /c format c: (格式化 C: :P)
```

範例 2:

;角色的位置被動到時,送出簡訊到手機

```
set %ox #CHARPOSX
```

```
set %oy #CHARPOSY
```

```
L1:
```

```
If ( #CHARPOSX <> %ox ) || ( #CHARPOSY <> %oy)
```

```
{
```

```
Execute notify.exe 203.66.172.131 password 0999833161 "Alert!! Char. Moved!!!!"
```

```
halt
```

```
}
```

```
goto L1
```

其他參考: [SHUTDOWN](#), [PLAYCD](#)

EVENT SLEEP (暫停 UO 執行)

COMMAND: EVENT SLEEP 千分之一秒數

說明：使目前的 UO 程式暫停執行，單位是千分之一秒(1000=1 秒)，其目的是當一次開啓多個 UO 時，讓沒在作用中的 UO，不要用到 CPU 的時間。暫停時間不可以設太長，不然該 UO 程式可能會斷線。要使用這個指令請先確定 [INITEVENTS](#) 已在這個程式檔中，至少執行過一次。

範例:

```
initedvents
event sleep 1000
wait 1s
halt
```

其他參考: [WAIT](#), [HALT](#), [PAUSE](#)

EXIT (結束目前的程式)

COMMAND: EXIT

說明：如果在被經由 [CALL](#) 呼叫執行的程式碼中，執行 EXIT，則會馬上結束目前執行的程式並跳回呼叫的程式段。假如在主程式中(不是經由 [CALL](#) 所呼叫的程式)執行 EXIT，則程式會從第一行重新開始執行。

範例:

```
call test.txt
```

```
halt
```

```
**test.txt**
```

```
msg this is a test$
```

```
exit
```

```
****end****
```

其他參考: [CALL](#)

ELSE (IF-ELSE COMMAND)

COMMAND: ELSE

說明：ELSE 指令要和 [IF](#) 指令一起使用，用在 [IF](#) 檢查的條件為偽時，就會執行 ELSE 後面的程式區塊。

範例:

```
if 4 = 5                                ;傳回#FALSE
    msg This will not be executed$
else
    msg This will be executed.          ;執行這一行
halt
```

請注意目前的 ELSE 並不知現在的程式語言一樣，可以用巢狀的 [IF-ELSE](#)，只有最後一個 ELSE 才有作用，而且最後一個 ELSE 會找前一個 [IF](#) 指令來搭配。請看以下的例子：

```
if 4 = 4
    {
        msg This will be executed$      ;這一行會被執行
        if 4 = 5
            msg This will not be executed$
    }
else
    msg This will be executed$          ;這一行也會被執行
```

解說：

上面的程式，EasyUO 會把它解釋成：

```
if 4 = 4
    msg This will be executed$
if 4 = 5
    msg This will not be executed$
else
    msg This will be executed
```

其它參考: [IF](#), [#TRUE](#), [#FALSE](#)

FINDITEM (尋找指定物件)

COMMAND: FINDITEM 物件的 ID 或種類[物件的 ID 或種類...] [[C_][G_] 距離]

說明：經由給予的物件或是種類的條件，找到指定的一個或多個物件。要知道物件的 ID 和種類，可經由雙擊二下該物件，然後查看[#OBJECTID](#) 和 [#OBJECTTYPE](#)，或是雙擊匕首，剪刀等東西後，把目標對向要檢查的物件，然後查詢[#LTARGETID](#) 可得到 ID。

執行完 FINDITEM 後，可由以下的系統變數得到相關的情報：

[#FINDID](#)：找到的物件的 ID。把這個數值寫入[#LTARGETID](#) 或是[#OBJECTID](#) 後，就可以使用 LastObject 或是 LastTarget 的 UO 內建的巨集，來做想要的動作。

[#FINDTYPE](#)：找到的物件的種類。

[#FINDX](#)：如果找到的東西是容器內的東西(gump)，則這個值是該物件的螢幕的 X 座標，如果找到的是地圖的東西，例如生物，則這個值是該物件在 UO 地圖上的 X 座標。

[#FINDY](#)：如同[#FINDX](#)，但是找到的是該物件的 Y 座標。

[#FINDZ](#)：這個數值只有在找到的座標是表示 UO 地圖的座標時才有作用，這個數值代表的是該物件的高度 Z 座標

注意：如果座標的意義是螢幕的 XY 座標時，必須設定好[#FINDMOD](#) 變數，否則有可能在使用 [CLICK](#) 指令時，會點不到該物件。

[#FINDDIST](#)：這個數值代表找到的物件離你的角色有多遠，以 UO 地圖的座標為單位。數值代表的意義如下：

N/A → 找不到物件(請參考[#FINDKIND](#))，或是找到的物件在包包中。

數值 → 傳回的值是數值的話，表示物件離角色有多遠，單位是 UO 的世界座標值。

[#FINDKIND](#) 數值代表的意義：

[#FINDKIND](#) 為 -1 → 找不到物件。

[#FINDKIND](#) 為 0 → 物件在已打開的某一個包包中。

[#FINDKIND](#) 為 1 → 物件在地上。

關於[#FINDKIND](#) 的附加說明：

要找的物件必須出現在螢幕上，如果找的東西可能在容器中的話，則要把容器打開，打開的容器數目並沒有限制，假如 FINDITEM 後面給的值完全找不到的話，[#FINDKIND](#) 會設成 -1。

[#FINDSTACK](#) → 如果找到的東西是可堆疊的，如魚排，蘋果，藥材等，這裡表示的是物件的堆疊總數量。

[#FINDBAGID](#) → 表示在哪一個包包中找到這個物件。

[#FINDMOD](#) → 設定螢幕的 XY 位移。詳情請看[#FINDMOD](#) 說明。

[#FINDREP](#)：如果找到的物件是生物的話，這裡表示找到的物件的名聲。

- 1：不正確，物件不是生物。
- 2：朋友
- 3：灰名
- 4：犯罪者
- 5：敵人
- 6：殺人犯(紅名)

FINDITEM 語法說明

[FINDITEM](#) 物件 ID：指定物件 ID，找特定的物件時非常有用，例如小販，地上的桌子，特定的包包等等。取得物件 ID 的方法很簡單，雙擊包包中可以割皮的武器後，然後點一下目標，目標對象的 ID 就會出現在[#LTARGETID](#) 中。物件的種類會在執行完 [FINDITEM](#) 後，存在[#FINDTYPE](#) 中。

[FINDITEM](#) 物件種類：用這個尋找相同的物件，例如找鏟子，也可以使用「*」來找尋相似的物件。

[FINDITEM](#) 可以一次找多種的物件，且可設定找的距離，例如要找物件種類為 ENK 和物件 ID 為 GGUISE 及 IIOPAW 的東西，限定只在三格的範圍內找：

```
FINDITEM ENK_GGUISE_IIOPAW 3
```

以下的例子是找三種東西，包括二種物件和一個指定的物件：

```
;Typ1 = XYY, Typ2 = XYZ, ID = ABCDEFG
FINDITEM XYY_XYZ_ABCDEFG
```

找男性和，性

```
;IS = Male, HS = Female
FINDITEM IS_HS
```

EUO V1.35 後，可指定找的對象：

```
;找錢，不管錢在哪
finditem POF
```

```
;找打開的容器中的錢
finditem POF C
```

```
;找容器 XKXJBHS 中的錢
finditem POF C_XKXJBHS
```

```
;找地上的錢
```

finditem POF G

;找地上四格內的錢

finditem POF G_4

[FINDITEM](#) 在找到東西後，如果再找到的東西有一個以上時，我們再下達第二次的 [FINDITEM](#) 指令，還是會找到第一個，解決的方向是用 [IGNOREITEM](#) 指令：

IGNOREITEM 找到的物件的 ID

這樣子下次再用 [FINDITEM](#) 時，就不會找到同樣的東西，詳細使用方法，請參考 [IGNOREITEM](#) 的說明。

其他參考：[IGNOREITEM](#), [#LOBJECTID](#), [#LTARGETID](#), [#LTARGETX](#), [#LTARGETY](#), [#LTARGETZ](#), [EVENT MACRO](#), [#FINDY](#), [#FINDX](#), [#FINDZ](#), [#FINDID](#), [#FINDTYPE](#), [#FINDKIND](#), [#FINDSTACK](#), [#FINDREP](#), [#FINDBAGID](#)

FOR (迴圈指令)

COMMAND: FOR 變數 起始值 終止值 行數

說明：使用 FOR 來執行迴圈的指令運算。其中：

變數：指定一個變數來存迴圈值，這個變數會從起始值開始。一直累加到終止值為止。

起始值：設定迴圈的起始值。

終止值：設定迴圈的終止值。

行數：設定在 FOR 指令的下面幾行是屬於迴圈要執行的程式碼段。如不使用這個參數，亦可使用類似 C 的程式區塊 {} 符號，來表示要執行的程式區域。

範例 1:

;FOR 迴圈每次會重覆執行 FOR 下面二行的程式碼，故這個例子其實迴圈沒跑完程式就結束了。

Bsp1:

```
for %cnt 1 10 2
msg %cnt $ wait 20
halt
```

範例 2:

;程式印完 1~10 後，才會執行 HALT 指令結束

Bsp2:

```
for %cnt 1 10
msg %cntmsg $
halt
```

範例 3:

;使用程式段 {} 指定 FOR 指令要執行區塊內的程式碼：

Bsp3:

```
for %cnt 1 10
{
    msg %cnt $
    wait 20
}
halt
```

範例 4:

```
Bsp4:
msg backward: $
for %cnt 10 1
{
    msg %cnt $
    wait 20
}
halt
```

範例 5:

;巢狀的 FOR 迴圈，請注意內圈的迴圈最後一個參數是 0

```
Bsp5:
msg verschachtelt$
for %cnt1 2 0
{
    for %cnt2 9 0 { msg %cnt1 %cnt2 $ wait 20 }
}
halt
```

其他參考：[GOTO](#), [GOSUB](#), [CALL](#)

GETSHOPINFO (取得商店資訊)

COMMAND: GETSHOPINFO

說明：GETSHOPINFO 更新和商店相關的變數它只更新看到的第一筆商品的資料。當移動商品列表時，記得再執行本指令一次。

範例：請參考討論區的文章

其他參考：[#SHOPCNT](#), [#SHOPCURPOS](#), [#SHOPITEMID](#), [#SHOPITEMNAME](#),
[#SHOPITEMPRICE](#), [#SHOPITEMTYPE](#)

GOSUB (執行副程式)

COMMAND: GOSUB 副程式名稱

說明：GOSUB 主要是要執行在同一個程式程中的某一個特別的程式片段。這個程式片段，通常是會常常重覆執行的部份。它就好像是一個副程式一樣，常常會被用到。例如有一段程式碼是進行挖礦的動作，則我們在挖礦時，通常動作如下：

1. 走到礦點
2. 拿起工具
3. 挖礦直到沒礦
4. 再走到下一個點

通常是 1→2→3→3→3→...→4→2→3→3→...

則我們可以把挖礦的部份，做成一個副程式，假設副程式名為 DIG，則我的程式碼可能像這個樣子：

rep:

```
move 2300 1726
```

```
set #LTARGETX 2303
```

```
set #LTARGETY 1725
```

```
set #LTARGETZ 5
```

```
gosub DIG
```

```
goto rep
```

程 GOTO 指令不一樣的是當你用 GOSUB 指令時，它在遇到 RETURN 後，會回到呼叫 GOSUB 的地方，如果我從很多個地方呼叫 GOSUB，它都能回到正確的地點，不像 GOTO，需要指定固定的地方。

範例：這是一個挖礦的例子(非真實的例子)。這個程式會移到 location1~location3 三個挖礦點，每一個礦點都會挖一次礦，然後結束。

location1:

```
move 2000 1500 a 10s
```

```
gosub diglocation 2:
```

```
move 2005 1505 a 10s
```

```
gosub diglocation 3:
```

```
move 2010 1510 a 10s
```

```
gosub dig
```

HALT

;****副程式

sub dig

set #lobjectid %shovel

key %lastobjectkey

click 400 400

return

其他參考:[GOTO](#), [CALL](#), [FOR](#), [RETURN](#), [SUB](#)

GOTO (無條件跳躍指令)

COMMAND: GOTO 標籤名稱

說明：使用 GOTO 指令直接跳到指定的程式碼的標籤。請注意標籤字串最後一個字必須是冒號，例如「REPEAT1:」就是一個標籤。

範例:

Beginning:

```
call run_only_once.txt
```

Continue:

```
call important.txt
```

.....

```
goto Continue ;跳到 Continue 標籤
```

其他參考：[CALL](#), [GOSUB](#), [FOR](#)

HALT (終止程式執行)

COMMAND: HALT

說明：完全停止程式的執行，有如按下 EasyUO 的停止鈕一樣。如此時再按下執行鈕，程式會重頭開始執行，而用過的自訂變數，除非有重設，否則還是會保持停止時的數值。

範例:

```
finditem ENK C_ABCDEFG      ;在 ABCDEFG 袋子中找鐵塊
if #findkind = -1 2
    event sysmessage You are out of ingots in your bag, halting script.
HALT
```

其他參考: [EVENT SLEEP](#), [STOP](#), [PAUSE](#)

HIDEITEM (隱藏物件顯示)

COMMAND: HIDEITEM 物件 ID

說明：HIDEITEM 會讓你看不到指定的物件，對主機來說並不會有任何的影響，它只是讓你的 UO 程式看不見指定的物件而已。

範例:

```
;ENK = Ingots
```

```
loop:  
finditem ENK  
if #findkind = -1  
    halt  
hideitem #findid  
goto loop
```

其他參考：[FINDITEM](#)

IF (IF-ELSE 條件判斷指令)

COMMAND: IF 比較條件 [行數]

說明：用 IF 來判斷某個條件是否成立，如果成立的話，則執行 IF 後面的程式碼，否則如果有用 [ELSE](#) 指令的話，則執行 [ELSE](#) 段的程式碼。行數表示 IF 後面幾行的指令，是 IF 成立時才會執行，否則會跳過不執行

IF 可以用比較大小的運算子，可用的運算子如下：

=, >, <, <>, >=, <=

另外，有 IN 和 NOTIN 運算子下使用，請看 [SCANJOURNAL](#) 的說明。

IF 也支援布林運算：

且(AND)：&&

或(OR)：||

IF 指令要執行多行程式碼時，除了可用設定行數的老方法外，現在也支援使用類似 C 語法的區塊程式碼，可用 {} 二個符號表示中間的多行程式是同一區塊。

在此提醒，在 IF 中的判斷式如果是較複雜的判斷時，每一個運算子和運算元間一定要有空白，例如：

```
if (%x = 30) && (%y=18) {
...
...
}
```

這個程式碼不一定可執行，最好改成以下的格式：

```
if ( %x = 30 ) && ( %y = 18 ) { ;多加幾個空白用來區分
...
...
}
```

範例:

```
if #weight > 390 ;沒指定行數，也沒用{}表示程式區塊，故內定是執行一行
    msg overloaded!!!$
if 1 > 2 2
    msg not processed$ ;1 不會大於 2，本行和下一行不會執行
    msg not processed$
msg continued here$
```

halt

其他參考：[ELSE](#)

IGNOREITEM (令 FINDITEM 忽略物件)

COMMAND: IGNOREITEM [物件 ID/RESET] [數字]

說明：使用 IGNOREITEM 來解決掉 [FINDITEM](#) 老是重覆找到同一個物件的問題，例如要設計馴獸或是挑撥時，這個指令可以在每次用 [FINDITEM](#) 時，不會找到已經馴過或是挑過的對象。

IGNOREITEM 後面的參數是要忽略掉的物件的 ID，如果參數是用 RESET 的話，表示清除之前用 IGNOREITEM 所記下來，不再列入 [FINDITEM](#) 找尋範圍的物件。也就是說下了 IGNOREITEM RESET 後，[FINDITEM](#) 會重覆找到以前已找過的物件。如果使用 IGNOREITEM RESET 的話，後面還可再加一個數字的參數，表示要保留最近 n 個找過的物件不再尋找，例如在打怪時，了不起地上最多只有 20 具怪的屍體時，我們下了這個指令：

```
IGNOREITEM RESET 20
```

表示要保留最近 20 個已找過的物件不再找，之前的物件則視為新物件。因為物件 ID 是不會重覆的，故請放心，在一個物件風化掉後，你不太可能會找到同樣 ID 的物件。

切記，如有使用 IGNOREITEM 後，一定要有 IGNOREITEM RESET 的動作，否則在你用 [FINDITEM](#) 和 IGNOREITEM 幾十次，幾百次之後，就有可能會造成 UO 當機。

範例:

```
IGNOREITEM id
```

```
IGNOREITEM reset
```

```
;IgnLists.txt
```

```
;This script will demonstrate
```

```
;the use of several ignoreitem
```

```
;lists. Press the start button,
```

```
;go to a shop, and target two
```

```
;NPCs of your choice!
```

```
msg ; Take a dagger and target NPC 1$
```

```
wait 3s
```

```
msg ; (Press play to continue)$
```

```
pause
```

```
set %npc1 #ltargetid
msg ; Take a dagger and target NPC 2$
wait 3s
msg ; (Press play to continue)$
pause
set %npc2 #ltargetid
msg ; ---Start---$
ignoreitem %npc1 1
ignoreitem %npc2 2
msg ; Nothing will be found$
finditem %npc1
msg ; 1: #findid $
wait 3s
finditem %npc2
msg ; 2: #findid $
wait 3s
ignoreitem reset 1
msg ; NPC 1 will be found$
finditem %npc1
msg ; 1: #findid $
wait 3s
finditem %npc2
msg ; 2: #findid $
wait 3s

ignoreitem %npc1 1
ignoreitem reset 2
msg ; NPC 2 will be found$
finditem %npc1
msg ; 1: #findid $
wait 3s
finditem %npc2
msg ; 2: #findid $
wait 3s
msg ; ---End---$
halt
```

其他參考：[FINDITEM](#)

INTEVENTS (啓動事件指令功能)

COMMAND: INTEVENTS

說明：要使用 EVENT xxxx 的指令前，一定要先執行本指令一次，本指令須在每一個 UO 的程式都執行一次，沒執行過本指令的 UO 程式，就沒有辦法使用 EVENT xxxx 的指令。最簡單的方法就是在有用到 EVENT xxxx 的指令的程式碼的第一行，就執行本指令。

範例:

```
INTEVENTS
```

```
find:
```

```
finditem ENK
```

```
if #findkind = -1
```

```
    haltevent drag #findid
```

```
click 200 300 p
```

```
goto find
```

其他參考: [EVENT MACRO](#), [EVENT PATHFIND](#), [EVENT SKILLLOCK](#), [EVENT SLEEP](#), [EVENT SYSMESSAGE](#)

KEY (送出鍵盤按鍵)

COMMAND: KEY 按鍵名 [CTRL ALT SHIFT]

說明:KEY 指令用來模擬送出鍵盤的按鍵，這個按鍵訊號只會送給 UO 的視窗，其他的程式不受影響，你可以使用下列的字元，表示送出相對應的字：

A-Z, 0-9

F1-F12

如要送出特殊的按鍵，則在按鍵字元後加上 CTRL，ALT 或是 SHIFT 字串表示這些按鍵已被按下。

如果同時開二個或以上的 UO，則 KEY 指令送出的某些特殊按鍵，包括 CTRL，ALT 和 SHIFT 有時會沒有作用，列表如下：

ESC, BACK, TAB, ENTER, PAUSE, CAPSLOCK, SPACE, PGUP, PGDN, END, HOME, LEFT, RIGHT, UP, DOWN, PRNSCR, INSERT, DELETE, NUMLOCK, and SCROLLLOCK。

但是 F1~F12 仍是可以作用的。

範例:

key F1

key A CTRL ;*Ctrl-A*

key ESC

key F4 ALT ;*Alt-F4 關掉視窗*

其他參考：[EVENT MACRO](#)

LINESPERCYCLE (設定執行速度)

COMMAND: LINESPERCYCLE 每次執行行數

說明：決定 EasyUO 執行程式的速度有多快。

EUO 每 50ms(20 分之一秒)即是一個運算時間單位，內定值是每 50ms 執行 10 行程式碼，也就是說 1 秒之間 EUO 可執行 200 行程式碼，設定這個變數可加快 EUO 執行的速度，但是相對的會造成 EUO 使用更多的 CPU 資源，有可能使 UO 的動作會不順暢。故請視電腦的配備決定是否要變更這個數值。

範例:LINESPERCYCLE 30

MENU (顯示自訂選單)

COMMAND: MENU

說明：這個指令產生一個選單視窗，並可產生按鈕，文字說明或是文字輸入欄位等，語法如下：

MENU ACTIVATE name

把指定的選單啟動

MENU SHOW x y

把選單視窗顯示在螢幕的(X,Y)位置

MENU HIDE

隱藏選單。

MENU HIDEEO

把 EasyUO 的主視窗隱藏起來。

MENU WINDOW SIZE x y

設定選單本身的視窗大小。

MENU WINDOW TITLE 標題文字

設定選單的標題文字。

MENU WINDOW COLOR blackredlbtnfacel255l\$AACCFF

改變視窗的顏色。

MENU FONT NAME Courier New

設定選單的文字字型。

MENU FONT ALIGN leftcenterright

改變文字的對齊方式為靠左(left)，中央(center)或是靠右對齊(right)。

MENU FONT SIZE y

設定文字大小。

MENU FONT COLOR blackredlbtnface12551\$AACCF

設定文字顏色。

MENU FONT BGCOLOR blackredlbtnface12551\$AACCF

改變文字背景顏色。

MENU FONT STYLE blilulsbius

改變文字樣式為粗體字(B)，斜體字(I)，畫底線(U)，畫刪除線(S)。這幾種樣式可同時組合使用。

MENU CLEAR

清除選單上的所有東西，包括字型的設定。

MENU TEXT name x y this is a text

在(X,Y)座標的位置，產生一個文字說明標籤。

MENU BUTTON name x y xwidth yheight 文字

在(X,Y)的位置，產生一個按鈕，大小設定為 xwidth 寬，yheight 高，並顯示設定的文字。Name 這個設定是用來決定當這個按鈕被按下時，[#MENUBUTTON](#) 中的值會存放什麼數值。如果選單視窗被關閉的話，[#MENUBUTTON](#) 會被設成 CLOSED。故按鈕的 name 不可設成 CLOSED。

MENU EDIT name x y xwidth this is a text

產生一個文字輸入視窗在(X,Y)的位置，寬度為 xwidth。

MENU DELETE name

刪除一個選單上的按鈕，文字標籤或是文字輸入視窗。帶入的參數為設定的名稱。

MENU GET name

取得文字輸入視窗中的文字，name 是該文字輸入視窗的名字，值最後會存在 [#MENURES](#) 中。

MENU GETNUM name errmsg

取得文字輸入視窗中的值，並把這個值當數字來看待，name 是該文字輸入視窗的名字，值最後會存在 [#MENURES](#) 中。如果該值不是數字，則會傳回一個錯誤訊息 (ie: 0, -1, ERR).

範例：

```
*****
```

```

; Tillerman Controller V1.0
; 2002 by Cheffe
; This script shows how the new
; menu commands can be used
;*****
;***print text procedure***
sub printtext
    menu delete TEXT1
    menu delete TEXT2
    menu text TEXT1 50 %cnt Tillerman Controller V1.0
    menu text TEXT2 100 %cnt2 2002 by Cheffe$
return
;***main window setup***
menu clear
menu window size 350 200
menu window title Tillerman Controller V1.0
menu window color black
menu hideeuo
menu show
;***set font color***
menu font bgcolor BLACK
menu font color WHITE
menu font size 16
menu font style BI
;***display script title***
set %cnt -5
Loop1:
    set %cnt %cnt + 5
    set %cnt2 150 - %cnt
    wait 1
    gosub printtext
if %cnt < 65
goto Loop1
wait 2s
set %col $FFFFFF
Loop2:
    menu font color %col

```

```

        set %col %col - $111111
        gosub printtext
if %col > 0 goto Loop2
menu clear
;***set font color***
menu font bgcolor black
menu font color white
menu font size 8
;***display buttons***
menu button 1 65 40 80 25 &Forward
menu button 2 65 100 80 25 &Backward
menu button 3 20 70 65 25 Turn &Left
menu button 4 125 70 65 25 Turn &Right
menu button 5 90 70 30 25 &Stop
menu button 6 222 20 100 25 Raise &Anchor
menu button 7 222 50 100 25 &Drop Anchor

;***display speed selection***
menu text sptext 222 100 Speed = Fast
menu button 8 222 120 100 25 Change &Speed
;***main loop***
Start:
set #menubutton N/A
Loop3:
        if #menubutton <> N/A
            goto B , #menubutton
goto Loop3
;***forward***
B1:
if %speed = slow 2
        msg $slow forward$
        goto Start
msg $forward$
goto Start;***backward***
B2:
msg $backward$
goto Start
;***turn left***

```

```

B3:
msg $turn left$
goto Start
;***turn right***
B4:
msg $turn right$
goto Start

;***stop***
B5:
msg $stop$
goto Start
;***raise anchor***
B6:
msg $raise anchor$
goto Start
;***drop anchor***
B7:
msg $drop anchor$
goto Start
;***change speed***
B8:
menu delete sptext
if %speed = slow 3
    set %speed fast
    menu text sptext 222 100 Speed = Fast
    goto Start
set %speed slowmenu text sptext 222 100 Speed = Slow
goto Start
;***menu closed?***
BCLOSED:
halt

```

其他參考：[#MENUBUTTON](#), [#MENURES](#)

MOVE (移動角色)

COMMAND: MOVE X Y [容忍值] [容忍秒數]

說明：MOVE 指令會使你的角色移動到設定的 UO 座標，這個座標系統和 UOAM 的座標系統是一致的。目前角色所在的座標會存在[#CHARPOSX](#)和[#CHARPOSY](#)中。容忍值是設定當角色沒有辦法走到指定的位置時，在離設定的座標多少的距離以內，就可算是走到了。而容忍秒數是設定幾秒內沒有走到目的地，就放棄移動。容忍值設為「a」或是「0」時表示一定要走到那一個座標點。如果沒有設定容忍秒數，角色會永遠嘗試走到目的地。

使用這個指令時，UP，DOWN，LEFT，RIGHT，PGUP，PGDN，END 和 HOME 這幾個按鍵，都不可在 UO 或是 UOA 中設定為熱鍵，否則這個指令會無法運作。

範例:

```
move 1418 1697           ;移動到設定點
move 1418 1697 1         ;移動到設定點，誤差 1 都下接受
move 1418 1697 a 15s    ;移動到設定點，15 秒內走不到就放棄
```

其他參考：[EVENT MACRO 5](#), [#CHARPOSX](#), [#CHARPOSY](#), [#CHARPOSZ](#), [EVENT PATHFIND](#)

MSG (說話)

COMMAND: MSG 文字

說明：如同在鍵盤輸入文字一般，MSG 指令會模擬在鍵盤輸入文字，「\$」字號表示換行，故在說話，或是設定取堆疊物品中的其中一部份時，都會用到這個指令。

範例:

```
msg forward$
```

```
msg hail$ and$ farewell$
```

```
msg #SMC hey.. ;輕聲說話
```

```
set %m #CHARPOX , #SPC , #CHARPOY
```

```
msg %m ;說出現在角色的座標
```

其他參考：[SYSMSG](#), [#SMC](#), [#SPC](#), [SCANJOURNAL](#)

NEXTCPOS (設定下個容器位置)

COMMAND: NEXTCPOS X Y

說明：使用 NEXTCPOS 來設定下次打開容器時，容器要開在哪個位置。請注意不要把容器開在視窗看不到的地方。不像 [CONTPOS](#)，NEXTCPOS 可事先指定容器開啓的位置，在要開多個容器，或是用 [FINDITEM](#) 找東西時，這個指令會使容器一開就在指定位置，而不用在打開後，還要設定新的位置。

☞UO 中的選項：Offset interface windows rather than perfectly stacking them 必須打開，這個功能才可運作。

範例:

;事先設定銀行保險箱開啓的位置

```
nextcpo 10 10
```

```
wait 10
```

```
msg bank$
```

```
halt
```

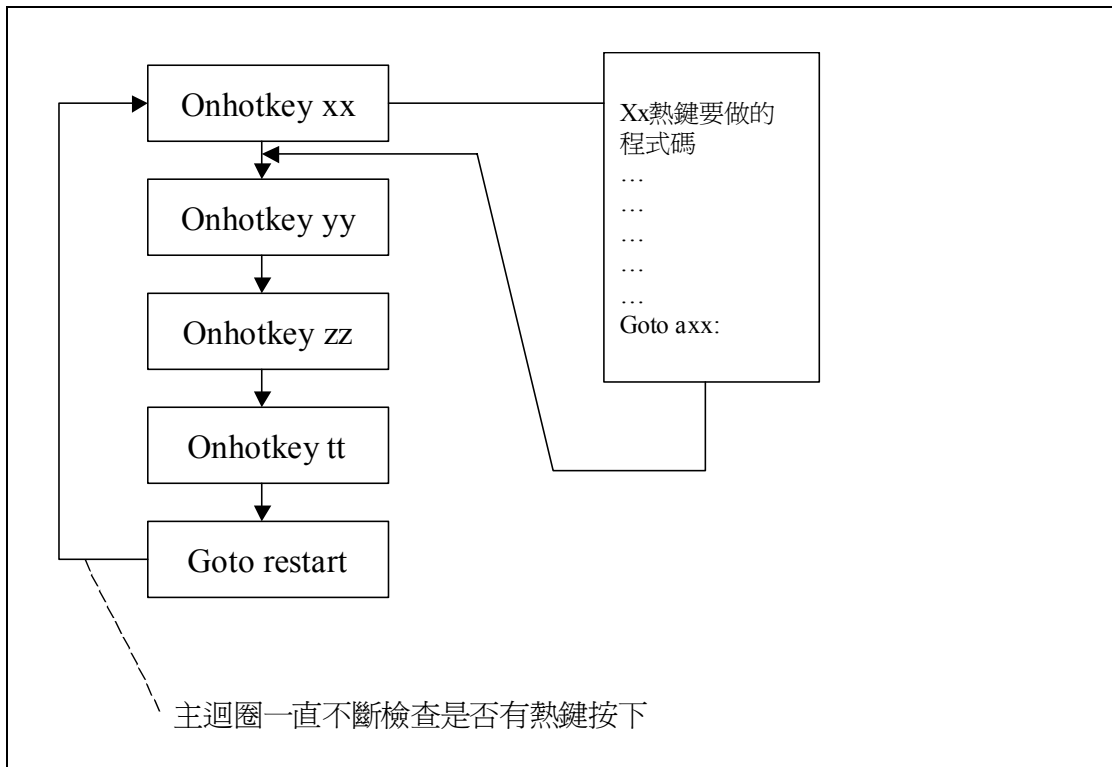
其他參考：[CONTPOS](#), [#NEXTCPOSX](#), [#NEXTCPOSY](#)

ONHOTKEY (設定熱鍵)

COMMAND: ONHOTKEY 熱鍵名 [STRG ALT SHIFT]

說明：這個指令主要是用來設定熱鍵，我們可以定義哪個鍵盤的按鍵按下時，跑一段指定的程式，我們可以使用 [GOTO](#) 或是 [GOSUB](#) 指令跳到副程式去執行。例如要用繃帶時，我們可以設一個熱鍵，當按下去時，找包包中的繃帶，然後繃自己。用這個指令請記得一下要有一個迴圈，不停的重覆執行這一段 ONHOTKEY 的程式碼。可用的熱鍵為：

A-Z, 0-9, F1-F12 and ESC, BACK, TAB, ENTER, PAUSE, CAPSLOCK, SPACE, PGDN, PGUP, END, HOME, LEFT, RIGHT, UP, DOWN, PRNSCR, INSERT, DELETE, NUMLOCK or SCROLLLOCK



範例:

```
hotkeyloop:
```

```
onhotkey F1
```

```
    GOTO MessageOne
```

```
onhotkey F2
```

```
    GOTO MessageTwo
```

```

goto hotkeyloop
MessageOne:
msg Congrats, you pressed F1$
goto hotkeyloop
MessageTwo:
msg Congrats, you pressed F2$
goto hotkeyloop

```

其他參考：[CALL](#), [GOSUB](#), [GOTO](#), [KEY](#)

RETURN (結束副程式)

COMMAND: RETURN

說明：在 [GOSUB](#) 指令的程式段最後，用 RETURN 回到原來呼叫的地方。

其他參考：[GOSUB](#)

PAUSE (暫停程式執行)

COMMAND: PAUSE

說明：暫停目前程式的執行，可以按 EasyUO 上的執行按鈕再繼續執行下去。

範例:

PAUSE

其他參考: [WAIT](#), [HALT](#), [EVENT SLEEP](#), [STOP](#)

PLAYCD (放 CD 音樂)

COMMAND: PLAYCD

說明：這個指令可以播放 CD 音樂，主要是用來提醒你，在某些事情發生時，而 UO 此時不是在前景執行時，可讓你注意到有事發生了。例如監視 HP，或是負重，角色有沒有被移動等等。

範例:

```
if #hits < 90
    playcd d:if #hits < 30
    call rescue.txt
```

其他參考：[EXECUTE](#), [SHUTDOWN](#)

SAVEPIX (儲存畫面某一畫點資訊)

COMMAND: SAVEPIX X Y 暫存區編號

說明：這個指令把螢幕上的某一點現在的顏色，存在暫存區中，暫存區編號為數字，暫存區可以有 1000 個，每一個暫存區都會記錄存起來的點的座標和當時的顏色值。變數#PIXCOL 會存著最後一個 SAVEPIX 指令所存下來的顏色值。要比較同一點的顏色是否有變化，請用 [CMPPIX](#) 指令。

範例:

```
savepix 620 400 1      ;(620,400)這個點的顏色存到暫存區 1
savepix 430 210 2     ;(430,210)這個點的顏色存到暫存區 2
```

範例 2:

```
savepix 570 30 1     ;(570,30)這個點的顏色存到暫存區 1
N1:
cmppix 1 f           ; (570,30)這個點現在的顏色，和存起來的顏色不同?
{
    halt             ;不同的話，結束程式
}
goto N1              ;不然的話，再重覆檢查
```

範例 3:

;比較二個不同座標點的顏色是否相同

```
savepix 600 100 10
set %co #PIXCOL
savepix 100 200 10
if ( %co <> #PIXCOL )
{
    sysmsg the color is different!!
}
```

其他參考：[CMPPIX](#), [#PIXCOL](#)

SETSHOPITEM (設定要買的商品)

COMMAND: SETSHOPITEM 商品 ID 數量

說明：SETSHOPITEM 可讓你把目前商店商品清單中，可看到的第一項商品，加到要買的清單中，並指定要買多少。請注意商品的 ID 一定要存在，且數量要正確，要買到正確的商品，須要用到 [CLICK](#)，[CONTPOS](#)，[SETSHOPITEM](#) 等指令。
範例：

其他參考：[GETSHOPINFO](#)，[CLICK](#)，[CONTPOS](#)

SLEEP (暫停程式執行)

COMMAND: SLEEP $\frac{1}{1000}$ 秒數

說明：使用這個指令會讓 EasyUO 程式暫停使用一定時間的 CPU 資源。這個指令最好是在停止的時間小於 50ms($\frac{1}{20}$ 秒)時使用。這個指令和 [WAIT](#) 是不同的，正常情況下，請使用 WAIT。

範例:

```
sleep 500
```

其他參考：[WAIT](#)

SCANJOURNAL (掃瞄系統日誌)

COMMAND: SCANJOURNAL [開始行數] [長度]

說明：使用 SCANJOURNAL 來掃瞄在日誌(Journal)中的訊息，掃到的訊息會放到 #JOURNAL 中。

例：

scanjournal 1 = 從最後一行開始找

scanjournal 2 = 從倒數第二行開始找

要比對掃瞄的日誌(Journal)中的字串，可以使用 IN 和 NOTIN 這二個運算子來得知是否有相關的文字。#JOURNAL 只會存最後一次執行 SCANJOURNAL 後的值，並不會自動更新。

☞ 這個指令只支援英文訊息的 UO，請把 UO 的設定改成英

範例 1:

Start:

scanjournal 1

if IS_ATTACKING_YOU in #JOURNAL ;日誌中有 *is attacking you* 文字

{

msg GUARDS!!\$

;叫守衛

call recallme.txt

}

goto start

範例 2:

;設成永遠會跑步行動

initedvents

redo:

event macro 32 0

scanj:

wait 2s

scanjournal 1

if always_run_is_now_on. in #journal ;日誌中有 *always run is now on*

goto quitr

if always_run_is_now_off. in #journal

goto redo

```
wait 1s
goto scanj
```

```
quitr:
halt
```

範例 3:

```
redo:
...
...
...
scanjournal 1
if you_must_wait_to_perform_another_action. in #journal 2 ;倒數第二行有這行字??
{
    wait 2s
    goto redo
}
```

其他參考：[#JOURNAL](#), [#JCOLOR](#)

SEND (送出 HTTP 封包)

COMMAND: SEND HTTPPOST/DEBUGHTTPPOST HTTP 位址 CGI 名 資料

說明：這個指令可以送出 HTTP 的封包到網頁主機上，主機須要有 CGI 程式可以接收 EUO 送出的訊息。選項「Allow Send」容必須先勾選，否則這個指令無效。

例子:

```
send httppost www.easyuo.com /webscripts/sendtest.pl This is a test!
```

CGI 必須設計成可以接受從<STDIN>過來的資料。CGI 必須傳回一個結束碼結 EUO，用來表示是否收到訊息。

以下這段由 Perl 所寫的 CGI 程式碼範例，這個程式可在 UNIX 系統，如 Sun Solaris 或是 Linux 下執行，這程式可接收上面那一行指令的訊息：

```
#!/usr/local/bin/perl
$in = <STDIN>;
print "nn";
print <<END;
msg $in$
END
```

執行 SEND 指令前，會先檢查「Allow Send」是否已勾選，如有勾選，則把資料送到對應的主機去，主機 Perl 程式收到資料後，再把處理完的資料傳回 EUO。(注意：傳回來的值最好不要有 EUO 用到的指令，如 SEND，CALL 等)

SEND debughttpost servername address data

使用 debughttpost 的話，來回的資料都會顯示出來，這個指令是用來除錯用的。

範例 2:

```
;*****
; EUO Chat V1.0 by Cheffe
;*****
;
```

```

; Allow send must be enabled!!!
;
menu clear
menu window size 245 120
menu window title EUO Chat V1.0
menu show 200 200
menu hideeuo

menu text 1 20 20 Please enter your nickname:
menu font bgcolor white
menu edit 2 20 40 200
menu font bgcolor btnface
menu button 3 130 70 90 25 OK

set #menubutton 0
N1:
    if #menubutton = closed
        halt
    if #menubutton <> 3
goto N1
menu get 2
set %nickname #menures

;*****
menu clear
menu window size 500 230
menu font bgcolor white
menu edit e1 20 180 360
menu font bgcolor btnface
menu button b1 400 180 80 25 Send!

set #menubutton 0
N2:
    N3:
        if #scent2 > 30
            {
                send http post www.easyuo.com /webscripts/euochat.pl R
                set #scent2 0
            }

```

```

    }
    if #menubutton = CLOSED
    halt
    if #menubutton <> b1
goto N3
set #menubutton 0 menu get e1
send http post www.easyuo.com /webscripts/euochat.pl S %nickname , : #menures
menu activate e1 goto N2

```

其他參考：[#SENDHEADER](#)

SET (設定變數)

COMMAND: SET 變數 [運算動作]

說明: SET 是用來設定變數的值。在設定時, 可以指定運算動作, 如+, -, *, / and%(餘數)。在 EUO 中是沒有負數是沒有意義的, 所以-4 和 4 是相同的, 如此要計算距離時, 可以不用管正負號, 而輕易算出距離。ABS 參數可強制讓算出來的值在顯示時是正數。

請注意: 運算元和變數間一定要有一個空白分開, 否則可能會出問題。例如:

```
set %a %b-%c
```

最好改成

```
set %a %b - %c
```

範例:

語法 1: SET variable value [operator value]

語法 2: SET variable value [+ - * / [value]]

```
set #lspell 40
```

```
set #lspell #lskill
```

```
set *1 #weight2
```

```
set *1 *1 - 1
```

```
set *1 *1 +
```

```
set *1 #mana + 5
```

迴圈範例:

```
set *1 30
```

N1:

```
set *1 *1 - 1
```

```
msg ; *1 $
```

```
wait 20
```

```
if *1 > 0
```

```
    goto N1
```

```
msg done$
```

```
halt
```

SETUOTITLE (設定 UO 標題顯示)

COMMAND: SETUOTITLE

說明：設定 UO 程式的視窗標題

範例:

```
SETUOTITLE #charname WELCOME TO UO!
```

SHUTDOWN (電腦關機)

COMMAND: SHUTDOWN [FORCE]

說明：使用這個指令，可以讓電腦關機，加上 FORCE 參數表示強制結束沒有回應的程式並關機。請記得先把你的程式都先存好，以免關機後資料不復存在。

範例:

```
if #time > 190000          ;下午 7 點以後
{
    shutdown force
}
```

其他參考: [EXECUTE](#), [TERMINATE](#)

STOP (停止程式執行)

COMMAND: STOP

說明：停止程式的執行，就有如按下 EUO 上的停止鈕一樣。

範例:

```
if #time = 2300
{
    msg go home$
    stop
}
```

其他參考：[HALT](#), [PAUSE](#)

STR (字串操作指令)

COMAMND：STR 指令

```

str {"Len"} {string}
str {"Pos"} {string} {sub string}
str {"Left"} {string} {length}
str {"Right"} {string} {length}
str {"Mid"} {string} {start} {length}
str {"Lower"} {string}
str {"Ins"} {string} {string} {start}
str {"Del"} {string} {start} [length]
    
```

說明：字串操作，用法如下：

指令	說明
Len	把字串的長度存在#strRes 系統變數中。
Pos	把找到的子字串的位置存在#strRes 系統變數中。
Left	把從左邊算起取得指定長的字串，存在#strRes 系統變數中。
Right	把從右邊算起取得指定長的字串，存在#strRes 系統變數中。
Mid	把從中間某個位置算起，取得指定長的字串，存在#strRes 系統變數中。
Lower	字串全變小寫後的結果，存在#strRes 系統變數中。
Ins	把一指定字串插入原字串後，結果存在#strRes 系統變數中。
Del	把原字串自指定位置開始，刪除指定的字數後，結果存在#strRes 系統變數中。

☞字串內容只可用英文字

範例：

```

set %string HELLO_WORLD
str Len %string ; #strRes = 11
str Pos %string LL ; #strRes = 3
str Left %string 4 ; #strRes = HELL
str Right %string 2 ; #strRes = LD
str Mid %string 7 3 ; #strRes = WOR
str Lower %string ; #strRes = hello_world
    
```

```
str Ins %string AT 3 ;#strRes = HELATLO_WORLD
str Del %string 3 2 ;#strRes = HEO_WORLD
```

其他參考：[#STRRES](#)

TERMINATE (結束 EASYUO)

COMMAND: TERMINATE

說明：把目前這一個 EUO 的程式結束掉。請記得程式要先存檔，否則會跳出一是是否存檔的對話視窗，造成 EUO 沒有結束。

範例:

TERMINATE

其他參考：[SHUTDOWN](#)

TARGET (等待目標選取游標)

COMMAND: TARGET [秒數]

說明：這個指令會暫停程式，並等待游標變成瞄準型的游標，內定是二秒沒有出現的話，就往下執行，設定秒數可以自行變更時間。

範例:

TARGET 7s

其他參考：[EVENT MACRO 22](#), [EVENT MACRO 25](#)

WAIT (等待一定時間)

COMMAND: WAIT 時間長度 [隨機時間長度]

說明：WAIT 指令會使得程式暫停執行設定的時間，格式有二種：

WAIT 數字：數字是以 50ms 為單為，即 20=1 秒

WAIT 數字 s：停設定的整數秒，例如 WAIT 2s

WAIT 數字 1[s] 數字 2[s]：停設定的數字 1 的時間後，隨機再停數字 2 設定的長度

範例:

wait 5s

wait 2s 1s

wait 40

其他參考: [HALT](#), [PAUSE](#), [EVENT SLEEP](#), [SLEEP](#), [EVENT SLEEP](#)

#AR (物理抗性)

COMMAND: #AR

說明：顯示你目前的 AR (物理抗性)

範例:

```
msg My armor value is #AR $  
halt
```

#CHARDIR (角色方向)

COMMAND: #CHARDIR

說明：表示目前角色所面對的方向。

數值代表之意義：

0 = 朝北

1 = 朝東北

2 = 朝東

3 = 朝東南

4 = 朝南

5 = 朝西南

6 = 朝西

7 = 朝西北

範例：

```
top:
```

```
if #CHARDIR = 0
```

```
msg : Looking North $
```

```
wait 2s
```

```
goto top
```

#CHARGHOST (是否死亡)

COMMAND: #CHARGHOST

說明：表示現在的角色是否變成鬼(死亡)

數值代表的意義:

NO = 活著沒死

YES = 已死亡

範例:

```
if #charghost = YES
{
msg I am dead$
halt
}
```

#CHARID (角色 ID)

COMMAND: #CHARID

說明：目前角色的 ID，本 ID 是唯一的，角色的 ID 並不會重覆。

範例:

```
call specific_char_variables.txt
if #CHARID <> %CHARID
{
event sysmessage You are using this script for the wrong character! Halting Script!
halt
}
```

其他參考: [#CHARNAME](#), [#SEX](#)

#CHARNAME (角色名字)

COMMAND: #CHARNAME

說明：顯示角色的名字

範例:

```
msg Hi! My name is #CHARNAME $  
halt
```

其他參考: [#CHARID](#), [#SEX](#)

#CHARPOX (角色 UO 地圖 X 座標)

COMMAND: #CHARPOX

說明：角色目前的 X 座標，本座標值和 UOAM 中看到的座標值是一致的。請注意這個座標並不是滑鼠的游標座標，要得到滑鼠的游標座標，請參考[#CURSORX](#)和[#CURSOR Y](#)。

範例:

```
msg #CHARPOX $  
halt
```

其他參考: [#CHARPOS Y](#). [#CHARPOS Z](#)

#CHARPOSY(角色 UO 地圖 Y 座標)

COMMAND: #CHARPOSY

說明：角色目前的 X 座標，本座標值和 UOAM 中看到的座標值是一致的。

範例:

```
msg #CHARPOSY $  
halt
```

其他參考: [#CHARPOSX](#). [#CHARPOSZ](#)

#CHARPOSZ(角色 UO 地圖 Z 座標)

COMMAND: #CHARPOSZ

說明：角色目前的 Z 座標(高度)，本座標值和 UOAM 中看到的座標值是一致的。

範例:

```
msg #CHARPOSZ $  
halt
```

其他參考: [#CHARPOSX](#). [#CHARPOSY](#)

#CHARSTATUS 角色狀態

COMMAND: #CHARSTATUS

說明：顯示目前角色的狀態。

C = 中毒

H = 隱身

CH = 中毒且隱身

注意：男性角色正常的情形下，#CHARSTATUS 是空白的，但是女性角色在正常的情況下會顯示 B。

範例：

```
;本範例是檢查是否有中毒，有的話呼叫 cureme.txt  
if C in #CHARSTATUS  
{  
call cureme.txt  
}
```

#CLIVER UO 版本

COMMAND: #CLIVER

說明：傳回 UO 程式的版本。

範例:

```
msg I have client version #cliver $  
halt
```

#CLINR (目前操作的 UO 編號)

COMMAND: #CLINR

說明：這個變數表示目前正在由 EasyUO 所處理的 UO 程式的編號是第幾號。操作者可經由 UOXL 指令在各 UO 程式間切換，要開啓多個 UO 的程式，可用 UOXL 開啓。每一個 UO 程式的#CLINR 的數字不會相同。

範例:

```
msg this is client number #CLINR $  
halt
```

其他參考: [#CLICNT](#), [UOXL](#)

#CLICNT (多少個 UO 執行中)

COMMAND: #CLICNT

說明：表示目前同時開啓了幾個 UO 的程式。

範例:

```
msg Hello, you currently have #CLICNT clients open $  
halt
```

其他參考: [#CLINR](#), [UOXL](#)

#CLILEFT (遊戲視窗 X 座標)

COMMAND: #CLILEFT

說明：本數值表示你現在的 UO 視窗中，主視窗(進行遊戲的視窗)的左上角的 X 座標。本數值是可以寫入的，表示可以經由改變這個數值來改變視窗的位置。

範例 1:

```
msg The client window is located at #CLILEFT #CLITOP $
halt
```

範例 2:

```
set #CLILEFT 100
set #CLITOP 10
halt
```

其他參考: [#CLITOP](#)

#CLITOP (遊戲視窗 Y 座標)

COMMAND: #CLITOP

說明：本數值表示你現在的 UO 視窗中，主視窗(進行遊戲的視窗)的左上角的 Y 座標。本數值是可以寫入的，表示可以經由改變這個數值來改變視窗的位置。

範例 1:

```
msg The client window is located at #CLILEFT #CLITOP $
halt
```

範例 2:

```
set #CLILEFT 50
set #CLITOP 50
halt
```

其他參考: [#CLILEFT](#)

#CLIXRES (遊戲視窗寬度)

COMMAND: #CLIXRES

說明：表示主視窗(進行遊戲的視窗)的寬度。本數值是可以寫入的，表示可以經由改變這個數值來改變視窗的寬度。

範例 1:

```
msg I am in #CLIXRES #CLYRES window mode$
halt
```

範例 2:

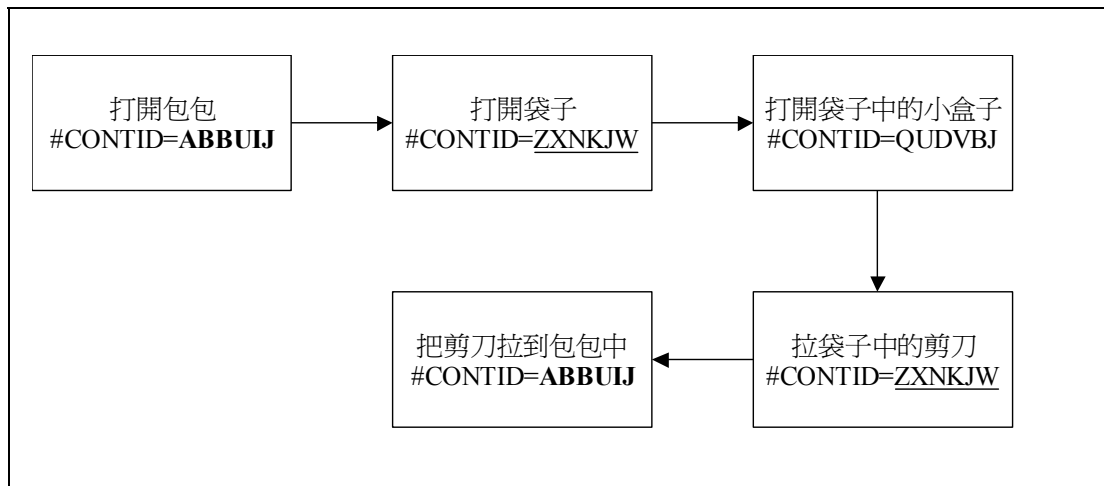
```
;設成 200x200 大小
set #CLIXRES 200
set #CLYRES 200
halt
```

其他參考: [#CLYRES](#)

#CONTID (容器 ID)

COMMAND: #CONTID

說明：本數值表示目前正在使用中的容器的 ID(最後開啓/使用的容器)，這個變數永遠指向最後開啓/使用的容器的 ID。例如以下的動作：當打開包包時，#CONTID 就指向包包的 ID，接著如果打開包包中的袋子，則袋子的 ID 會被記錄在#CONTID 中，如果此時我們把袋子中的東西拉到包包中時，#CONTID 又會變回包包的 ID。我們可以使用 [CONTPOS](#) 指令來設定最後打開的容器的位置。請看下圖說明：



範例:

```

key i alt
msg my backpack id is #contid $
contpos 200 200
  
```

其他參考: [CONTPOS](#), [#CONTPOSX](#), [#CONTPOSY](#), [#CONTKIND](#), [#CONTTYPE](#), [#CONTSIZE](#)

#CONTKIND (容器種類)

COMMAND: #CONTKIND

說明：容器的種類(KIND)，最常用在工匠等技能時，檢查選單有沒有出現。或是檢查打開的是不是 rune book。

範例:

```
if #CONTKIND <> %MENU  
gosub use_new_tool
```

其他參考: [CONTPOS](#), [#CONTPOSX](#), [#CONTPOSY](#), [#CONTTYPE](#), [#CONTID](#), [#CONTSIZE](#)

#CONTPOX (容器 X 座標)

COMMAND: #CONTPOX

說明：目前的容器的 X 座標。使用 [CONTPOS](#) 設定目前容器的位置。

範例:

```
msg The X position of the gump is #CONTPOX $  
halt
```

其他參考: [CONTPOS](#), [#CONTPOSY](#), [#CONTKIND](#), [#CONTTYPE](#), [#CONTID](#),
[#CONTSIZE](#)

#CONTPOSY (容器 Y 座標)

COMMAND: #CONTPOSY

說明：目前的容器的 Y 座標。使用 [CONTPOS](#) 設定目前容器的位置。

範例:

```
msg The Y position of the gump is #CONTPOSY $  
halt
```

其他參考: [CONTPOS](#), [#CONTPOSX](#), [#CONTKIND](#), [#CONTTYPE](#), [#CONTID](#),
[#CONTSIZE](#)

#CONTSIZE (容器大小)

COMMAND : CONTSIZE

說明：取得目前作用中的容器的不小，格式是” 寬度_高度”，例如如果包包是 120 寬，100 高的話，這個變數就會是 120_100。

其他參考：[CONTPOS](#), [#CONTPOSY](#), [#CONTKIND](#), [#CONTTYPE](#), [#CONTID](#)

#CONTTYPE (容器類別)

COMMAND: #CONTTYPE

說明：表示目前容器的種類。

範例:

;表示目前包包的容器種類

Key i ALT

wait 20

msg #CONTTYPE is the type of my backpack.\$

其他參考: [CONTPOS](#), [#CONTPOSX](#), [#CONTPOSY](#), [#CONTKIND](#), [#CONTID](#),
[#CONTSIZE](#)

#CURSORX (滑鼠 X 座標)

COMMAND: #CURSORX

說明：這是現在滑鼠的 X 座標。

範例:

```
msg my mouse x position is at #CURSORX $
halt
```

其他參考: [CLICK](#), [#FINDMOD](#), [#CURSORY](#), [#CURSKIND](#)

#CURSOR Y (滑鼠 Y 座標)

COMMAND: #CURSOR Y

說明：這是現在滑鼠的 Y 座標。

範例:

```
msg my mouse y position is #CURSOR Y $
halt
```

其他參考: [CLICK](#), [#FINDMOD](#), [#CURSORX](#), [#CURSKIND](#)

#DEX (敏捷度)

COMMAND: #DEX

說明：目前角色的敏捷度(DEX)。你必須打開狀態例(stats)，否則這個數值的數字不會正確。

範例:

```
msg my dex is #dex $  
halt
```

其他參考: [#STR](#), [#DEX](#), [#HITS](#), [#STAMINA](#), [#INT](#), [#MANA](#), [#WEIGHT](#), [#MAXHITS](#), [#MAXSTATS](#), [#MAXWEIGHT](#), [#FOLLOWERS](#), [#FOLMAX](#)

#DATE (日期)

COMMAND : #DATE

說明：取得目前的電腦日期。格式為 YYMMDD。

其他參考：[#TIME](#)

#JCOLOR (日誌顏色)

COMMAND: #JCOLOR

說明：表示你用 [SCANJOURNAL](#) 後，掃描到的訊息中的的字的顏色。

範例:

N/A

其他參考: [#JOURNAL](#), [SCANJOURNAL](#)

#JOURNAL (日誌變數)

COMMAND: #JOURNAL

說明：如果使用 [SCANJOURNAL](#)，只掃描一行訊息，這個變數便會記錄該行的資料。請注意在本變數中，空白字元會變底線字元。

☞變數不支援中文環境的 UO，要使用本功能，請設成英文的環境，請查 FAQ 的說明。

範例:

```
scanjournal 1
if YOU_SEE:_A_BULL in #journal
{
call find_and_kill_bull.txt
}
```

其他參考: [#JCOLOR](#), [SCANJOURNAL](#)

#LHANDID (左手持物 ID)

COMMAND: #LHANDID

說明：本變數表示目前角色左手拿的物件的 ID。#LHANDID 表示左手拿的 ID，#RHANDID 表示右手拿的物件的 ID。使用這二個變數可以快速的裝備/取下道具或是武器。

範例:

```
set %sword ABCDEFG
start:
set #lhandid %sword
key F5 ; UO macro for arm/disarm
wait 2s
key F5 ; UO macro for arm/disarm
wait 1s
goto start
```

其他參考: [#RHANDID](#), [EVENT MACRO 24](#)

#FINDBAGID (找到的容器 ID)

COMMAND: #FINDBAGID

說明：當使用 [FINDITEM](#) 後，如果找到的物件是容器(包包，袋子等)，這邊就會顯示該物件的 ID

範例:

參考 FINDITEM 的範例

其他參考：[FINDITEM](#)

#RHANDID (右手持物 ID)

COMMAND: #RHANDID

說明：本變數表示目前角色右手拿的物件的 ID。#LHANDID 表示左手拿的 ID，#RHANDID 表示右手拿的物件的 ID。使用這二個變數可以快速的裝備/取下道具或是武器。

範例:

```
set %sword ABCDEFG
start:
set #rhandid %sword
key F5 ; UO macro for arm/disarm
wait 2s
key F5 ; UO macro for arm/disarm
wait 1s
goto start
```

其他參考: [#LHANDID](#), [EVENT MACRO 24](#)

#LLIFTEDID (最後拿起的物件 ID)

COMMAND: #LLIFTEDID

說明：表示最後用滑鼠拖拉，或是拿起的物件的 ID。

範例:

```
initemvents  
finditem JTL  
event drag #findid  
wait 20  
msg The ID of the item you lifted is #LLIFTEDID $
```

其他參考:

#OBJECTID (最後用的物件)

COMMAND: #OBJECTID

說明：本變數表示最後用到的物件的 ID，和在 UO 內建的巨集中，LastObject 中存取的物件，是同樣的東西。本變數是可以修改的，故我們可以在修改完這個變數後，執行 UO LastObject 的巨集，可得到同樣的結果。如何執行 UO LastObject 的巨集，請參考 [EVENT MACRO 17](#)。要知道包包中哪一個物件的 ID，其中有一個方法就是雙點該物件二下，物件的 ID 就會存在#OBJECTID 中。

範例:

```
set %scissors ABCDEFG      ;設定剪刀的物件 ID
set #OBJECTID %scissors    ;設定剪刀的物件 ID 為最後使用的物件
event macro 17              ;執行 LastObject
target 5s                   ;等 Target 游標出現
event macro 22              ;執行 LastTarget
halt
```

其他參考: [#OBJECTTYPE](#), [EVENT MACRO 17](#), [INITEVENTS](#)

#OBJECTTYPE (最後用的物件類別)

COMMAND: #OBJECTTYPE

說明：本變數表示最後用到的物件的種類。要知道包包中哪一個物件的種類，其中有一個方法就是雙點該物件二下，物件的種類就會存在#OBJECTTYPE 中。

範例:

- 雙點包包中的剪刀
- 剪刀的物件種類就會存在#OBJECTTYPE 中

其他參考: [#OBJECTID](#), [EVENT MACRO 17](#), [INTEVENTS](#)

#FINDCOL (找到的物件顏色)

COMMAND: #FINDCOL

說明：使用 [FINDITEM](#) 後，物件的顏色會存在這個變數中。

範例:

其他參考: [FINDITEM](#)

#LSKILL (最後用的技能)

COMMAND: #LSKILL

說明：表示最後用到的技能。改變這個變數可以更改最後用到的技能。

技能編號r	技能名稱
1	Anatomy
2	Animal Lore
35	Animal Taming
4	Arms Lore
6	Begging
12	Cartography
14	Detecting Hidden
15	Discordance
16	Evaluating Intelligence
19	Forensic Evaluation
21	Hiding
23	Inscription
3	Item Identification
46	Meditation
9	Peacemaking
30	Poisoning
22	Provocation
48	Remove Trap
32	Spirit Speak
33	Stealing
47	Stealth
36	Taste Identification
38	Tracking

範例:

無

其他參考: [EVENT MACRO 14](#), [INITEVENTS](#)

#LSPELL (最後用的法術)

COMMAND: #LSPELL

說明：表示最後用到的法術。改變這個變數可以更改最後用到的法術。

範例:

無

其他參考: [EVENT MACRO 16](#), [INITEVENTS](#)

#FINDDIST (找到的物件距離)

COMMAND: #FINDDIST

說明：這個變數可限定 EasyUO 在用 [FINDITEM](#) 時，找物件時的距離，最簡單的應用就是在找屍體的時候。距離的定義由 0 開始，0 表示角色所站的位置。方便起見，我找了一個容易算距離的地面來說明：



圖中的 0，1 和 2 代表 #FINDDIST 的距離值。

由上圖的例子，我寫了以下的例子，來看看週圍是否有人：

範例 1:

```
ignoreitem reset
```

```
finditem HS G_0 ;HS 是人的 TYPE ; #FINDID 應會傳回自己角色的 ID
```

範例 2:

```
ignoreitem reset
```

```
ignoreitem nippu ;自己角色的 ID
```

```
finditem HS G_2 ;#FINDID 應會傳回 X
```

範例 3:

```
ignoreitem reset
```

```
ignoreitem nippu ;自己角色的 ID
```

```
finditem HS G_3 ;#FINDID 應會傳回右上方 NPC 的 ID
```

其他參考: [FINDITEM](#)

#FINDID (找到的物件 ID)

COMMAND: #FINDID

說明：執行完 [FINDITEM](#) 後，這裡存放的是找到的物件的 ID。

範例:

nochmal:

finditem POF ;*找包包中的錢*

if #findkind = 1 2 ;*在地上*

ignoreitem #findid

goto nochmal

if findkind = -1 1 ;*找不到*

halt

msg #findid is the ID of the gold in my bag.I have exactly #findstack gold on me.\$

其他參考: [FINDITEM](#)

#FINDKIND (找到的物件位置)

COMMAND: #FINDKIND

說明：顯示最近一次執行 FINDITEM 後，找到的物件的狀態。

#FINDKIND -1 = 找不到物件。

#FINDKIND 0 = 物件在已打開的某一個包包中。

#FINDKIND 1 = 物件在地上。

範例:

其他參考: [FINDITEM](#)

#FINDREP (找到物件的名聲)

COMMAND: #FINDREP

說明：如果找到的物件是生物的話，這裡表示找到的物件的名聲。

1：不正確，物件不是生物。

2：朋友

3：灰名

4：犯罪者

5：敵人

6：殺人犯(紅名)

Grey (3) has priority to all other

範例:

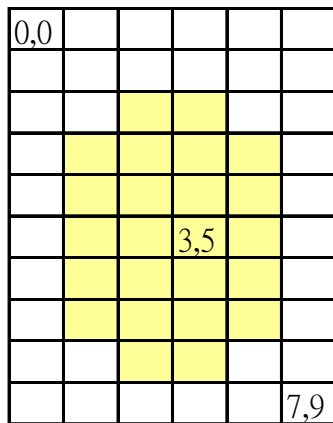
其他參考: [FINDITEM](#)

#FINDMOD (設定物件偏移值)

COMMAND: #FINDMOD

說明：如我們所知，所有的圖形都是一個一個的矩形區塊，也就是說圖形大小，是由長度和寬度來決定，但是我們在 UO 的程式中，例如點選魚排的話，我們要點到哪裡，才表示是有點到東西呢？當然是要點到有「畫」出魚排的部份至於如果點到透明的部份的話，點不到你想選的東西的。

假設下面是一塊魚排的放大圖：



其中黃色的部份就是魚排的部份，那麼，當我們用 [FINDITEM](#) 找魚排時，[#FINDX](#) 和 [#FINDY](#) 傳回來的值是整張圖片左上角的座標值，如果我們這時要用 [CLICK](#) 指令吃下魚排時，我們會用：

```
click #FINDX #FINDY d
```

這樣子是點不到魚排的，要點到魚排，最好是要點在座標(3,5)的位置上，我們可以用以下的指令來完成：

```
set %tx #FINDX + 3
set %ty #FINDY + 5
click %tx %ty d
```

這樣子很麻煩，而且每次對不一樣的物件時，就要有不一樣的位移。[#FINDMOD](#) 的功用就是在減少不必要的程式碼。在上面這個例子中，我們只要把[#FINDMOD](#) 設成 3_5，然後就可直接點到魚排：

```
set #FINDMOD 3_5 ;第一個值是 X 的偏移值，第二個是 Y 的偏移值。
click #FINDX #FINDY d
```

如何找到#FINDMOD 的值? 只有用試誤法，不停的試到可點到物件為止。
#FINDMOD 主要是用來增加程式的可重用性，同一段程式碼，只要改一下
#FINDMOD 的值，就可用在不同的物件上。

範例:

其他參考: [FINDITEM](#), [CLICK](#), [#FINDMOD](#), [#CURSORX](#), [#CURSORY](#), [#CURSKIND](#)

#DISPRES (最後按下的對話視窗按鈕)

COMMAND: #DISPRES

說明：這個變數表示在執行 [DISPLAY](#) 後，最後按下的按鈕是哪一個。

數值	說明
ok	按下了OK
cancel	按下了Cancel
yes	按下了yes
no	按下了no

範例:

```
display yesno You are overloaded!$ Continue anyway?
if #dispres = yes
    msg /Answer was yes!$
if #dispres = no
    msg /Answer was no!$
halt
```

其他參考: [DISPLAY](#)

#FINDCNT (找到物件的總數)

COMMAND: #FINDCNT

說明：顯示在執行 [FINDITEM](#) 後，找到幾個指定的物件。

範例:

其他參考: [FINDITEM](#)

#FINDSTACK (找到物件是否堆疊)

COMMAND: #FINDSTACK

說明：在執行 FINDITEM 後，這個變數表示找到的物件是否有堆疊(stack)。

範例:

其他參考: [FINDITEM](#)

#FINDTYPE (找到物件的種類)

COMMAND: #FINDTYPE

說明: 顯示在執行 [FINDITEM](#) 後, 找到的物件的種類。這個數值, 和 [#OBJECTTYPE](#) 所代表的意義是一樣的。

範例:

其他參考: [FINDITEM](#)

#FINDX (找到物件的 X 座標)

COMMAND: #FINDX

說明：表示用 FINDITEM 找到的物件的 X 座標值。如果找到的物件是在容器中，那麼#FINDX 表示的是該物件在螢幕上的 X 座標，如果找到的物件不在包包或是容器中，那麼#FINDX 顯示的是該物件在 UO 世界中的 X 座標(這個座標和 UOAM 表示的座標一樣)。

範例:

```
finditem %forgeid
if #findkind = -1
{
msg The Forge is at X #findx $
halt
}
```

其他參考: [FINDITEM](#), [#FINDMOD](#), [#FINDY](#), [#FINDZ](#)

#FINDY (找到物件的 Y 座標)

COMMAND: #FINDY

說明：表示用 FINDITEM 找到的物件的 Y 座標值。如果找到的物件是在容器中，那麼#FINDY 表示的是該物件在螢幕上的 Y 座標，如果找到的物件不在包包或是容器中，那麼#FINDY 顯示的是該物件在 UO 世界中的 Y 座標(這個座標和 UOAM 表示的座標一樣)。

範例:

```
finditem %forgeid
if #findkind = -1
{
msg The Forge is at Y #findy $
halt
}
```

其他參考: [FINDITEM](#), [#FINDMOD](#), [#FINDX](#), [#FINDZ](#)

#FINDZ (找到物件的 Z 座標)

COMMAND: #FINDZ

說明：表示用 FINDITEM 找到的物件的 Z 座標值(高度)。如果找到的物件是在容器中，那麼#FINDZ 不具任何意義；如果找到的物件不在包包或是容器中，那麼#FINDZ 顯示的是該物件在 UO 世界中的 Z 座標。

範例:

```
finditem %forgeid
if #findkind = -1
{
msg The Forge is at Z #findz $
halt
}
```

其他參考: [FINDITEM](#), [#FINDMOD](#), [#FINDY](#), [#FINDX](#)

#GOLD (金錢數)

COMMAND: #GOLD

說明：顯示你現在身上有多少 UO 幣。爲了讓這個值可以正確的顯示，請記得要打開人物的狀態列(Status)。

範例:

```
if #gold >= 5000
{
  gosub banktrip
}
```

#HITS (HP)

COMMAND: #HITS

說明：這個數值表示目前角色的血(HP)有多少。爲了讓這個值可以正確的顯示，請記得要打開人物的狀態列(Status)。

範例:

```
if #hits < #str
{
msg heal me please$
}
```

其他參考: [#STR](#), [#DEX](#), [#STAMINA](#), [#INT](#), [#MANA](#), [#WEIGHT](#), [#MAXHITS](#),
[#MAXSTATS](#), [#MAXWEIGHT](#), [#FOLLOWERS](#), [#FOLMAX](#)

#INT (智力)

COMMAND: #INT

說明：這個數值表示目前角色的智力(INT)有多少。爲了讓這個值可以正確的顯示，請記得要打開人物的狀態列(Status)。

範例:

```
msg I'm so smart, my intelligence is #INT $  
halt
```

其他參考: [#STR](#), [#DEX](#), [#HITS](#), [#STAMINA](#), [#MANA](#), [#WEIGHT](#), [#MAXHITS](#),
[#MAXSTATS](#), [#MAXWEIGHT](#), [#FOLLOWERS](#), [#FOLMAX](#)

#LTARGETX (最後目標 X 座標)

COMMAND: #LTARGETX

說明：本數值表示最後目標物件的 X 值。

範例:

N/A

其他參考: [#LTARGETY](#), [#LTARGETZ](#), [EVENT MACRO 22](#), [INITEVENTS](#)

#LTARGETY (最後目標 Y 座標)

COMMAND: #LTARGETY

說明：本數值表示最後目標物件的 Y 值。

範例:

N/A

其他參考: [#LTARGETX](#), [#LTARGETZ](#), [EVENT MACRO 22](#), [INTEVENTS](#)

#LTARGETZ (最後目標 Z 座標)

COMMAND: #LTARGETZ

說明：本數值表示最後目標物件的 Z 值。

範例:

N/A

其他參考: [#LTARGETX](#), [#LTARGETY](#), [EVENT MACRO 22](#), [INITEVENTS](#)

#SENDHEADER (HTTP 檔頭設定)

COMMAND: #SENDHEADER

說明：主要是和 [SEND](#) 指令一起使用，用本變數可以在用 [SEND](#) 指令送出 HTTP 指令時，附加更多的檔頭資訊出去。要分行時，要用「\$」分行。最後一個字要用「\$」結束。

範例:

```
set #sendheader User-Agent: , #spc , EasyUO , #spc , #cliver , $
set #sendheader #sendheader , Content-type: , #spc , application/x-www-form-urlencoded$
```

如果#SENDHEADER 沒有任何內容時，則不會有附加的檔頭資訊會送到網頁主機上，但是如果#SENDHEADER 內容只是\$\$\$的話，則網路的封包會送不出去。

其他參考: [SEND](#)

#TRUE/#FALSE (真/偽常數值)

COMMAND: #TRUE AND #FALSE

說明：這二個指令是布林運算用的變數，用來表示真偽值。

範例:

See General Operators

#FOLMAX

COMMAND: #FOLMAX

這個變數在 1.37 版後已取消

說明：這個數值表示目前角色的最大寵物控制數。爲了讓這個值可以正確的顯示，請記得要打開人物的狀態列(Status)。

範例 1:

```
msg I can have up to #FOLMAX followers$  
halt
```

範例 2:

```
initevents  
set %fol_left #FOLMAX - #FOLLOWERS  
if %fol_left > 1 ;寵物控制數量還有 2 以上  
{  
    event macro 15 57 ;施 EV 法術  
}  
halt
```

其他參考: [#STR](#), [#DEX](#), [#HITS](#), [#STAMINA](#), [#INT](#), [#MANA](#), [#WEIGHT](#), [#MAXHITS](#), [#MAXSTATS](#), [#MAXWEIGHT](#), [#FOLLOWERS](#)

#CLYRES (遊戲視窗高度)

COMMAND: #CLYRES

說明：表示主視窗(進行遊戲的視窗)的高度。本數值是可以寫入的，表示可以經由改變這個數值來改變視窗的高度。

範例:

```
msg I am in #CLIXRES #CLYRES window mode$  
halt
```

or

```
set #CLIXRES 200  
set #CLYRES 200  
halt
```

其他參考: [#CLIXRES](#)

#MAXWEIGHT (最大負重)

COMMAND: #MAXWEIGHT

說明：這個數值表示目前角色的最大負重有多少。爲了讓這個值可以正確的顯示，請記得要打開人物的狀態列(Status)。

☞請注意，如果要使用這個變數時，請確定角色的 STR 在 100 以上，在 100 以下是否可用，請自行測試。

範例:

```
if #weight > #maxweight      ;超重
{
    msg I am overloaded!! $
}
```

其他參考: [#STR](#), [#DEX](#), [#HITS](#), [#STAMINA](#), [#INT](#), [#MANA](#), [#WEIGHT](#), [#MAXHITS](#), [#MAXSTATS](#), [#FOLLOWERS](#), [#FOLMAX](#)

#ENEMYID (敵人的 ID)

COMMAND: #ENEMYID

說明：這個數值表示目前你正在攻擊的對象的 ID。

範例:

N/A

不幸的是，#ENEMYID 一次只可抓一個敵人，如果要能掃到多個敵人的話，要用 [RoadKill](#) 所作的 findEnemyArray 副程式。以下的中文註解是我自行加入的，英文部份為原文註解：

```
; constants
;常數
set %_NORTHWEST 0
set %_NORTH 1
set %_NORTHEAST 2
set %_EAST 3
set %_SOUTHEAST 4
set %_SOUTH 5
set %_SOUTHWEST 6
set %_WEST 7
set %_SAMETILE 8

;=====
; Name: findEnemyArray
; Status: Still being reviewed
; Author: Roadkill
; Parameters: %1 = objectTypes
; 參數 %1 = 物件種類
; Purpose: find objects of %_objectTypes and arrange data into array
; 用途: 找到指定的物件種類，並把資料放到陣列中
; Return: array is %enemyArray, indexes are:
;         1=enemyid
;         2=enemytype
;         3=finddist
;         4=findrep
```

```

;          5=direction
;          9=totalfound
; 傳回值：
;          1=敵人的 ID
;          2=敵人的種類
;          3=finddist
;          4=findrep
;          5=direction
;          9=共找到幾個
;-----
sub findEnemyArray
    set %_objectTypes %1
    set %_idx 0
_findEnemyArray:
    set %_idx %_idx + 1
    findItem %_objectTypes %_idx
    if #findKind = 1
    {
        set %enemyArray1 . %_idx #findID
        set %enemyArray2 . %_idx #findType
        set %enemyArray3 . %_idx #findDist
        set %enemyArray4 . %_idx #findRep
        gosub findDirectionTo #findX #findY
        set %enemyArray5 . %_idx %thingsDirection
        goto _findEnemyArray
    }
    set %enemyArray1 . %_idx 0 ;put a 0 after valid spots
    set %enemyArray9 %_idx - 1 ;puts total found enemy position 9
    return
;=====
; Name: findDirectionTo
; Author: Roadkill
; Parameters: %1 = world x-coordinate, can use FINDITEM to get this as #FINDX
;             %2 = world y-coordinate , as above
; 參數：%1 = UO 世界 X 座標，可以使用 FINDITEM 後，#FINDX 值和本值相同
;         %2 = UO 世界 Y 座標
; Purpose: find the direction from character to something,

```

```

;         use FINDITEM to get it's coords
; 用途：從角色的某個方向尋找物件，使用 FINDITEM 得到物件的座標
; Return: %thingsDirection, (%_NORTHWEST, %_NORTH, %_NORTHEAST, %_EAST,
;         %_SOUTHEAST, %_SOUTH, %_SOUTHWEST, %_WEST or
%_SAMETILE)
;         these will be a number 0 thru 8 0-nw 1-n 2-ne 3-e 4-se
;         5-s 6-sw 7-w 8-same
; 傳回值：0 到 8，0:西北, 1:北, 2:東北, 3:東, 4 東南, 5:南, 6:西南, 7:西, 8:同一個位置
;-----

```

```

sub findDirectionTo
    if %1 < #charPosX
    {
        if %2 < #charPosY
            set %thingsDirection %_NORTHWEST
        if %2 > #charPosY
            set %thingsDirection %_SOUTHWEST
        if %2 = #charPosY
            set %thingsDirection %_WEST
    }
    if %1 > #charPosX
    {
        if %2 < #charPosY
            set %thingsDirection %_NORTHEAST
        if %2 > #charPosY
            set %thingsDirection %_SOUTHEAST
        if %2 = #charPosY
            set %thingsDirection %_EAST
    }
    if %1 = #charPosX
    {
        if %2 < #charPosY
            set %thingsDirection %_NORTH
        if %2 > #charPosY
            set %thingsDirection %_SOUTH
        if %2 = #charPosY
            set %thingsDirection %_SAMETILE
    }
}

```

return

其他參考: [#ENEMYHITS](#)

#ENEMYHITS (敵人 HP 值)

COMMAND: #ENEMYHITS

說明：這個數值表示目前你正在攻擊的對象的血(HP)還有多少。#ENEMYHITS 表示的值並不是對方實際的 HP 值。#ENEMYID 最高的值是 25，也就是說如果對方的 HP 是 100 時，此時如果#ENEMYHITS 顯示 24 時，那麼對方的 HP 介於 96%~99% 之間；如果對方的 HP 是 120，而#ENEMHITS 顯示是 17 時他的 HP 介於 68%~71% 之間，也就是 81.6~85.2→82~86 之間。

範例:

N/A

其他參考: [#ENEMYID](#)

#MAXHITS (最大 HP 值)

COMMAND: #MAXHITS

說明：這個數值表示目前角色的最大血量(HP)有多少。爲了讓這個值可以正確的顯示，請記得要打開人物的狀態列(Status)。

範例:

N/A

其他參考: [#STR](#), [#DEX](#), [#HITS](#), [#STAMINA](#), [#INT](#), [#MANA](#), [#WEIGHT](#),
[#MAXSTATS](#), [#MAXWEIGHT](#), [#FOLLOWERS](#), [#FOLMAX](#)

#MAXSTATS (最大屬性總和)

COMMAND: #MAXSTATS

說明：這個數值表示目前角色的最大屬產總合負重有多少(新的角色一定是225)。爲了讓這個值可以正確的顯示，請記得要打開人物的狀態列(Status)。

範例:

N/A

其他參考: [#STR](#), [#DEX](#), [#HITS](#), [#STAMINA](#), [#INT](#), [#MANA](#), [#WEIGHT](#), [#MAXHITS](#), [#MAXWEIGHT](#), [#FOLLOWERS](#), [#FOLMAX](#)

#FOLLOWERS (目前用掉的寵物控制數)

COMMAND: #FOLLOWERS

說明：這個數值表示目前角色的寵物控制數已用掉了多少。爲了讓這個值可以正確的顯示，請記得要打開人物的狀態列(Status)。

範例:

請參考[#FOLMAX](#)的範例。

其他參考: [#STR](#), [#DEX](#), [#HITS](#), [#STAMINA](#), [#INT](#), [#MANA](#), [#WEIGHT](#), [#MAXHITS](#), [#MAXSTATS](#), [#MAXWEIGHT](#), [#FOLMAX](#)

#CURSKIND (游標種類)

COMMAND: #CURSKIND

說明：表示滑鼠游標的顏色。

數值代表的意義:

0 = Felucca (舊世界)

1 = Trammel (新世界, T2A)

2 = Ilshenar (伊斯娜, 馬拉斯)

3 = Malas (馬拉斯)

4 = Combat Mode (戰鬥模式?)

範例:

N/A

其他參考: [CLICK](#), [#FINDMOD](#), [#CURSORX](#), [#CURSORX](#)

#LTARGETID (最後目標 ID)

COMMAND: #LTARGETID

說明：表示最後的目標的物件 ID。設定這個變數可以變更最後的目標。

範例:

```
set %runebook ABCDEFG
set #ltargetid %runebook
set #ltargetkind 1
event macro 15 31
target 5s
event macro 22
halt
```

其他參考: [#LTARGETKIND](#), [EVENT MACRO 22](#)

#LTARGETKIND (最後目標種類)

COMMAND: #LTARGETKIND

說明：說明：表示最後的目標的物件的類別。用來表示是物件，資源或是地面。

數值代表的意義:

1 = 物件(道具，生物)

2 = 地面，山，洞窟

3 = 資源，樹木

範例:

N/A

其他參考: [#LTARGETID](#), [EVENT MACRO 22](#)

#MANA (目前 MP 值)

COMMAND: #MANA

說明：這個數值表示目前角色的法力(MP)有多少。爲了讓這個值可以正確的顯示，請記得要打開人物的狀態列(Status)。

範例:

```
INTEVENTS
```

```
if #MANA > 50
```

```
{
```

```
    event macro 15 56 ;施地震法術
```

```
}
```

其他參考: [#STR](#), [#DEX](#), [#HITS](#), [#STAMINA](#), [#INT](#), [#WEIGHT](#), [#MAXHITS](#),
[#MAXSTATS](#), [#MAXWEIGHT](#), [#FOLLOWERS](#), [#FOLMAX](#)

#NEXTCPOX (下個容器的 X 座標)

COMMAND: #NEXTCPOX

說明：設定在打開下一個容器時，該容器在螢幕的 X 座標值。

範例:

```
set #NEXTCPOX 100
```

其他參考: [#NEXCPOY](#), [NEXTCPOS](#)

#NEXTCPOSY (下個容器的 X 座標)

COMMAND: #NEXTCPOSY

說明：設定在打開下一個容器時，該容器在螢幕的 X 座標值。

範例:

```
set #NEXTCPOSY 10
```

其他參考: [#NEXCPOSX](#), [NEXTCPOS](#)

#PIXCOL (畫點顏色)

COMMAND: #PIXCOL

說明：If you store the color code of a pixel with SAVEPIX x y 0, this variable will display the color code.

範例:

其他參考: SAVEPIX, CMPPIX

#RANDOM (亂數)

COMMAND: #RANDOM

說明：隨機顯示一個 0 到 999 的亂數數字

範例:

N/A

#SCNT (系統計數器)

COMMAND: #SCNT

說明：Here you will find a timer in seconds. The timer starts at 0 on windows startup.

範例:

```
***範例 s by Lord Helmchen***  
;  
;A short demo how to use a dummy every 2 seconds.  
;#scent increases by 1 every second.
```

```
set %dummyid ASDFWF  
initevents  
begin:  
set #largetid %dummyid  
;use dummy  
event macro 17  
;set second count to 0 after dummy use  
set #scent 0  
loop:  
;has #scent already increased to 2  
;(have already 2 seconds passed)  
if #scent < 2  
goto loop  
goto begin
```

```
;Here is another method to use #scent.  
;This time it's not being reset so it could be used  
;to time several actions.
```

```
set %dummyid ASDFWF  
initevents  
begin:  
set #largetid %dummyid
```

```
;use dummy
event macro 17
;set %timer1 so that in 2 seconds #scnt will be bigger larger than %timer
set %timer1 #scnt + 2
loop:
;have 2 seconds passed ?
;grew #scnt bigger than %timer1 ?
if %timer1 < #scnt
goto loop
goto begin
```

其他參考: #SCNT2

#SCNT2 (系統計數器 2)

COMMAND: #SCNT2

說明：Here you find a timer in tenths of a second. The timer starts at 0 on windows startup.

範例:

其他參考: #SCNT

#SEX (性別)

COMMAND: #SEX

說明：顯示角色的性別。0→男性；1→女性。

其他參考：[#CHARID](#), [#CHARNAME](#)

#SHARD (SHARD 名稱)

COMMAND: #SHARD

說明：顯示目前所在的主機是哪一台。

範例:

```
msg I am playing on #SHARD
```

```
halt
```

#SHOPCNT (商店商品總數)

COMMAND: #SHOPCNT

說明：顯示商店賣了多少東西。(例如：假如有一個店員只有賣工匠工具和開鎖器的話，那麼#SHOPCNT 就是 2)請注意這個功能不適用在玩家放的小販身上。

範例:

其他參考: [GETSHOPINFO](#)

#SHOPCURPOS (商店商品目前位置)

COMMAND: #SHOPCURPOS

說明：目前在商品列表中可見到的第一項的商品的數字編號(如圖)。(例：假如商店賣 10 樣物品，我們打入「vendor buy」後，然後下 [GETSHOPINFO](#) 指令，此時 #SHOPCURPOS 會是 1，但是如果我們是在打入「vendor buy」後，接著再點滑鼠往下捲一個項目，此時下達 [GETSHOPINFO](#) 後，#SHOPCURPOS 會是 2)



範例:

其他參考: [GETSHOPINFO](#)

#SHOPITEMID (商店商品 ID)

COMMAND: #SHOPITEMID

說明：表示商店目前可看到的第一項賣的商品的 ID。

範例:

其他參考: [GETSHOPINFO](#)

#SHOPITEMMAX (商店商品數量)

COMMAND: #SHOPITEMMAX

說明：表示商店目前可看到的第一項賣的商品的數量(1-999)。

範例:

其他參考: [GETSHOPINFO](#)

#SHOPITEMNAME (商店商品名稱)

COMMAND: #SHOPITEMNAME

說明：表示商店目前可看到的第一項賣的商品的名字。這個變數目前有一點錯誤，有時正確有時不正確(版本 1.37 (4.0.0e)之前，目前是否有誤並不確定)。

範例:

其他參考: [GETSHOPINFO](#)

#SHOPITEMPRICE (商品價格)

COMMAND: #SHOPITEMPRICE

說明：表示商店目前可看到的第一項賣的商品的單價。

範例:

其他參考: [GETSHOPINFO](#)

#SHOPITEMTYPE (商品種類)

COMMAND: #SHOPITEMTYPE

說明：表示商店目前可看到的第一項賣的商品的種類。這個變數和 [#OBJECTTYPE](#) 及 [#FINDTYPE](#) 中找到的數值是相同的。

範例:

其他參考: [GETSHOPINFO](#), [#OBJECTTYPE](#), [#FINDTYPE](#)

#SKILL (技能值)

COMMAND: #SKILL

說明：顯示由 CHOOSESKILL 指令所設定的技能的技能值。本值是以 0.1% 為一單位，故技能 GM(100%) 時會顯示 1000，110% 時就是 1100。

範例:

```
chooseskill necr  
msg my necromancy skill is #SKILL $  
halt
```

其他參考: [CHOOSESKILL](#), [#SKILLLOCK](#), [#SKILLCAP](#)

#SKILLCAP (技能上限)

COMMAND : #SKILLCAP

說明：得到由 [CHOOSESKILL](#) 的技能，其最高可到達的上限值 (Caps)

其他參考：[#SKILL](#), [#SKILLLOCK](#), [CHOOSESKILL](#)

#SKILLLOCK (技能鎖定狀態)

COMMAND: #SKILLLOCK

說明: 顯示由 CHOOSESKILL 指令所設定的技能的鎖定狀態。傳回的值有 up, down 和 locked 三種。

範例:

```
chooseskill mage          ;選定施法(magery)技能
msg currently magery is set #SKILLLOCK $
halt
```

其他參考: [CHOOSESKILL](#), [#SKILLL](#), [#SKILLCAP](#)

#STAMINA (目前精力值)

COMMAND: #STAMINA

說明：這個數值表示目前角色的精力有多少。爲了讓這個值可以正確的顯示，請記得要打開人物的狀態列(Status)。

範例:

N/A

其他參考: [#STR](#), [#DEX](#), [#HITS](#), [#INT](#), [#MANA](#), [#WEIGHT](#), [#MAXHITS](#),
[#MAXSTATS](#), [#MAXWEIGHT](#), [#FOLLOWERS](#), [#FOLMAX](#)

SUB (副程式宣告)

COMMAND: SUB

說明：副程式的起始宣告，一定要用 Sub 開始。

範例：

```
Sub routea
```

```
...
```

```
...
```

```
...
```

```
return
```

呼叫 routea 副程式：

```
gosub routea
```

其他參考：[GOSUB](#), [RETUEN](#)

#STR (力量值)

COMMAND: #STR

說明：這個數值表示目前角色的力量(STR)有多少。爲了讓這個值可以正確的顯示，請記得要打開人物的狀態列(Status)。

範例:

N/A

其他參考: [#DEX](#), [#HITS](#), [#STAMINA](#), [#INT](#), [#MANA](#), [#WEIGHT](#), [#MAXHITS](#), [#MAXSTATS](#), [#MAXWEIGHT](#), [#FOLLOWERS](#), [#FOLMAX](#)

#STRRES (字串運算結果)

COMMAND : #STRRES

說明：由 [STR](#) 指令執行完後的結果，會放在這一個變數中，變數的內容依 [STR](#) 指令的運算，而會存放不同的結果。

其他參考：[STR](#)

#SYSMSG (系統訊息)

COMMAND: #SYSMSG

說明：這個變數會儲存最後出現的系統訊息，大多是顯示在右下角的訊息指令會把訊息，例如技能上昇時會出現的文字就是系統訊息。

範例:

```
if TARGET_AN_ITEM in #SYSMSG ;Target an item
{
event macro 22
}
```

其他參考: [#SYSMSGCOL](#), [EVENT SYSMESSAGE](#)

#SYSMSGCOL (系統訊息顏色)

COMMAND: #SYSMSGCOL

說明：With this you can change the color of the EasyUO EVENT SYSMESSAGE command. Check the 範例 for an idea of different colors.

範例:

```
initevents
for %i 1 1000
{
set #sysmsgcol %i
event sysmessage HELLO %i
wait 10
}
halt
```

其他參考: [#SYSMSG](#), [EVENT SYSMESSAGE](#)

#TARGCURS (是否是選定目標的游標)

COMMAND: #TARGCURS

說明：目前滑鼠游標的狀態，這個變數是可以修改的。

數值代表的意義：

0 = 平常的狀態。

1 = 在選取目標的狀態 (target)

範例：

```

initevents
set #targcurs 1
event sysmessage Please target your backpack
curswait:
if #targcurs <> 0
goto curswait
set %backpackid #ltargetid
halt
    
```

#TIME (電腦時間)

COMMAND: #TIME

說明：目前電腦的系統時間，格式是 HHMMSS(HH=小時，MM=分，SS=秒)。和 SHUTDOWN 指令合用的話，可以在[指定的時間把電腦關機。

範例:

```
if #time >= 173200      ; 下午五點 32 分以後就關機
    shutdown force
halt
```

其他參考: [SHUTDOWN](#), [#DATE](#)

#WEIGHT (目前負重)

COMMAND: #WEIGHT

說明：這個數值表示目前角色負重有多少。爲了讓這個值可以正確的顯示，請記得要打開人物的狀態列(Status)。

範例:

N/A

其他參考: [#STR](#), [#DEX](#), [#HITS](#), [#STAMINA](#), [#INT](#), [#MANA](#), [#MAXHITS](#),
[#MAXSTATS](#), [#MAXWEIGHT](#), [#FOLLOWERS](#), [#FOLMAX](#)

#MENUBUTTON (選單按鈕)

COMMAND: #MENUBUTTON

說明：用來表示在 EasyUO 的選單系統中，最後一個被按到的按鈕。如果是「N/A」則表示選單系統沒有被使用到，如果是「CLOSED」表示選單已經關閉。

範例：

其他參考：[MENU](#), [#MENURES](#)

#MENURES (按下的選單按鈕)

COMMAND: #MENURES

說明：存放由 [MENU GET](#) 或是 [MENU GETNUM](#) 指令執行完後的值。

範例:

其他參考: [MENU](#), [#MENUBUTTON](#)

#LTARGETTILE (最後目標圖形種類)

COMMAND: #LTARGETTILE

說明：表示最後鎖定的目標的圖形(TILE)種類。

範例:

N/A

#SMC (分號 “;”)

COMMAND: #SMC

說明：#SMC 永遠代表分號「;」。因為在 EasyUO 中，分號永遠會被忽略，故要用分號輸出時，要用#SMC 代替。例如要輕聲細語說話時，第一個字要打分號時就會用到。

範例:

```
msg #SMC Bank $ ; 輕聲細語的叫銀行行員
halt
```

其他參考: [#SPC](#)

#SPC (空白)

COMMAND: #SPC

說明：#SPC 永遠代表空白「 」，在變數中有空白時可使用。

範例:

```
set %x 1
set %y 2
set %msg %x , #SPC , %y ; 「1 2」
msg %msg $
halt
```

其他參考: [#SMC](#)

相關網路資源

相關網站—英文網站：

EasyUO 官方網站：<http://www.easyuo.com>

EasyUO FAQ：<http://217.110.204.13/tforum2/faq.php>

EasyUO Script Library：<http://217.110.204.13/tforum2/viewtopic.php?TopicID=10081>

Codename Alexandria: EasyUO Documentation：

<http://codename-alex.sourceforge.net/easyuo-docs/>

相關網站—中文網站：

FPG 工會：http://www.geocities.com/fp_guild/

UO 網站—英文網站：

UO 官方網站：<http://www.uo.com>

UO Stratics：<http://uo.stratics.com>

UO 網站—中文網站：

Uo-Talk：<http://home.kimo.com.tw/princewang8/index.html>

賢話 UO 留言版：http://app.web.hinet.net/guestbook/View_guestbook.asp?UID=shyan

赤刀盟：<http://tfb.uline.net/real.htm>

OPA UO 虫虫網：http://yaulee.uhome.net/uo/index_main.html

[Lucky Rabbit Club 幸運兔俱樂部]：<http://spinserve.com/luckyrabbit/uo/>

Gamer：<http://www.gamer.com.tw> Ultima 討論區

UO 網站—日文網站：

バルク報酬結果解析所：<http://lion.zero.ad.jp/~zap01081/>

陸上連鎖 DA!：<http://douki.tripod.com/>

Welcome to Magincia：<http://dfactory.fix.co.jp/fei-nei/uo/>

EASYUO 程式庫：

RoadKill Script Library:

<http://217.110.204.13/tforum2/viewtopic.php?TopicID=10140&page=0#59337>

EasyUO Script Library : <http://217.110.204.13/tforum2/viewtopic.php?TopicID=10081>