Forum:Hack Forums

Thread: Waiting for keys

annikk.exe - *June 29, 2015, 6:39 am*

Here's a script I made today.

Code
```
@echo off
setlocal EnableDelayedExpansion
REM Keyboard cursor prototype

call :createBuffer 40 30
call :createPlayer 0 0

set frame=0

:loop
   cls
   echo %buffer%
   set /a frame+=1
   title X: %playerX%   Y: %playerY%   FRAME: %frame%
   call :getKey
   call :moveChar %playerX% %playerY% %key%
goto :loop

pause
exit /b

:createPlayer
   call :insertChar %1 %2 "X"
   set playerX=%1
   set playerY=%2
goto :eof

:moveChar
   REM Moves a character.  Replaces the character at the previous position with a space.
   set oldX=%1
   set oldY=%2
   set k=%~3

   set newX=%oldX%
   set newY=%oldY%

   REM Which direction should the character move in?
   if /i %k% == W set /a newY-=1
   if /i %k% == A set /a newX-=1
   if /i %k% == S set /a newY+=1
   if /i %k% == D set /a newX+=1

   REM Is the new position out of bounds?  If so, take no action
```

```
    if %newX% GEQ %bWidth%     goto :eof
    if %newY% GEQ %bHeight% goto :eof
    if %newX% LSS 0     goto :eof
    if %newY% LSS 0     goto :eof

    REM Calculate new and old buffer indices
    set /a oldIndex=(%oldY% * %bWidth%) + %oldX%
    set /a newIndex=(%newY% * %bWidth%) + %newX%

    REM Capture the character to be moved
    set char=!buffer:~%oldIndex%,1!

    REM Adjust positions to correct for 0 indexing
    set /a oldIndexAdj=%oldIndex% + 1
    set /a newIndexAdj=%newIndex% + 1

    REM Replace the existing character with a space
    set "buffer=!buffer:~0,%oldIndex%! !buffer:~%oldIndexAdj%,%bufferSize%!"

    REM Insert the character in its new position
    set "buffer=!buffer:~0,%newIndex%!%char%!buffer:~%newIndexAdj%,%bufferSize%!"

    REM Update the Player with the new position
    set playerX=%newX%
    set playerY=%newY%
goto :eof

:insertChar
    REM Inserts a character into the buffer, replacing whatever char was there before
    set posX=%1
    set posY=%2
    set char=%3
    set /a charPos=(%posY% * %bWidth%) + %posX%
    set "buffer=!buffer:~0,%charPos%!%~3!buffer:~%charPos%,%bufferSize%!"
goto :eof

:getKey
    REM Gets the key presses from the user
    for /F "delims=" %%a in ('xcopy /l /w "%~f0" "%~f0" 2^>nul') do (
        set "key=%%a"
        goto :gotKey
    )
    :gotKey
    set "key=%key:~-1%"
    for %%a in (W A S D) do (if /i "%key%"=="%%a" goto :eof)
goto :getKey

:createBuffer
    REM Set up the buffer and screen mode, init some variables
    set bWidth=%1
    set bHeight=%2
    set /a bufferSize=%bWidth% * %bHeight%
    for /L %%a in (1,1,%bufferSize%) do (set "buffer=!buffer! ")
```

```
    set /a bHeight+=2
    mode %bWidth%,%bHeight%
    set /a bHeight-=2
goto :eof
```

The user presses the WASD keys to control a little "X". They can move the X around within the bounds of the screen.
The mode (screen size) can be changed and the script will adapt to this dynamically.

This shows some basics of buffers and so on, but that's not what I want to talk about today.

The problem with the approach outlined in this script is that it's impossible to have something continuing in the background while we wait for a key press.

For example, we might animate a "Z" character bouncing between the left and right walls. However, the animation will only advance a frame when the user presses a key. If the user stops pressing keys, the Z will not move.

How can we capture key presses in this way, but allow the script to continue running if no key was pressed? Developing a fast and elegant system that allows us to do this could potentially open the floodgates for a heap of cool batch games.
waiting for the miracle... for the miracle to come

WOLGA - *June 29, 2015, 11:11 am*
Okay, here is my solution:

Code @echo off

```
echo.>press

if not "%~1"==""" goto background
start "" /b /d %cd% cmd /c ""%~0" "background""

setlocal enabledelayedexpansion

for /f "tokens=1-4 delims=:.," %%a in ("!time: =0!") do set /a
"t1=(((1%%a*60)+1%%b)*60+1%%c)*100+1%%d-36610100"

set mover=1.1
set tmpA=1
set tmpB=1

for /l %%. in () do (

for /f "tokens=1-4 delims=:.," %%a in ("!time: =0!") do set /a
"t2=(((1%%a*60)+1%%b)*60+1%%c)*100+1%%d-36610100"

set /a tDiff=t2-t1
if !tDiff! lss 0 set /a tDiff+=24*60*60*100

if !tDiff! geq 10 (
set t1=%t2%
```

```
set "buffer="
for /l %%A in (1,1,25) do (
for /l %%B in (1,1,25) do (
set buffer=!buffer!!%%A.%%B!
))

mode con cols=25 lines=25
for /l %%A in (1,1,25) do (
for /l %%B in (1,1,25) do (
set "%%A.%%B= "
))

for /f %%A in (press) do (
set "tomove=%%A"
title !tomove!
)

for /f "tokens=1-2 delims=." %%A in ("!mover!") do (
if %%A lss 25 (
set /a tmpA=%%A+1
) else (
set tmpA=1
if %%B lss 25 (
set /a tmpB=%%B+1
) else (
set tmpB=1
)
)
)

set "!mover!= "
set mover=!tmpA!.!tmpB!
set "!mover!=X"

cls
echo !buffer!

)
)

:background

:getKey
for /f "delims=" %%K in ('xcopy /l /w "%~f0" "%~f0" 2^>nul') do (
    set "key=%%K"
    goto :next
)
:next
set "key=%key:~-1%"
if /i %key%==a echo a>press
if /i %key%==d echo d>press
goto getKey
```

Unfortunately, I could not get it working like I hoped, but you can see that when you are running this, the keys "A" and "D" will change the title, while not blocking the animation.

annikk.exe - *June 30, 2015, 12:23 am*
That's an interesting proof of concept, Qwerty.

These lines seem critical:
Code if not "%~1"==""" goto background
start """ /b /d %cd% cmd /c ""%~0" "background""
So on first run, we check for a parameter, and there isn't one.. The conditional basically says "go to background if parameter 1 is blank". On the first run, parameter 1 will definitely be blank since the user probably double-clicked the file from the Windows GUI.
So then it proceeds to the second line, where it starts itself and passes the word "background" as the first parameter.
This means when the newly launched process checks the conditional, this time there IS a parameter, which means we jump straight to the Background section and start outputting key presses to the "press" file.
Meanwhile, the initial process continues on after starting the background process, animates the display, and checks for a key press on each loop.

So, my question is how are we able to have both processes running within the same Window? I had no idea that was possible. It's groundbreaking..
I need to see some more examples of it working. I will try to code up some tests for it today.

Update:

Here's a simplified version:

Code @echo off
echo.X>catPress

if not "%~1"==""" goto :getKey
start """ /b /d %cd% cmd /c ""%~0" "CATS""

set numCats=0
:loop
    for /f %%a in ('type catPress') do (title %%a)
    set /a numCats+=1
    echo %numCats%
    timeout /t 1 /nobreak >nul
goto :loop

:getKey
    for /f "delims=" %%K in ('xcopy /l /w "%~f0" "%~f0" 2^>nul') do (
        set "key=%%K"
        goto :next
)
    :next
    set "key=%key:~-1%"
    if /i %key%==a echo a>catPress
    if /i %key%==d echo d>catPress
goto getKey
This accepts user inputs whilst running an "animation" in real time.

This is definitely possible, then.
I now need to:
1. Implement the new technique into the existing script
2. Create a reliable sub-second delay function
3. I'll have a shot at making that bouncing "Z"...

WOLGA - *June 30, 2015, 9:32 am*
Let me talk about your post.

Quote:The conditional basically says "go to background if parameter 1 is blank".
Exactly the opposite, I'm sure you meant that, but you should fix it in the post.

Quote:So, my question is how are we able to have both processes running within the same Window? I had no idea that was possible. It's groundbreaking..
Not sure myself about the "how", but it's a trick White taught me, so kudos to him.

Okay, so I got my code working like I was hoping it would.

Code @echo off

```
echo.>press

if not "%~1"==""" goto background
start "" /b /d %cd% cmd /c ""%~0" "background""

setlocal enabledelayedexpansion

mode con cols=25 lines=27

for /f "tokens=1-4 delims=:.," %%a in ("!time: =0!") do set /a
"t1=(((1%%a*60)+1%%b)*60+1%%c)*100+1%%d-36610100"

set mover=1.1
set tmpA=1
set tmpB=1

set char=5.5
set tmpAC=5
set tmpBC=5

for /l %%. in () do (

for /f "tokens=1-4 delims=:.," %%a in ("!time: =0!") do set /a
"t2=(((1%%a*60)+1%%b)*60+1%%c)*100+1%%d-36610100"

set /a tDiff=t2-t1
if !tDiff! lss 0 set /a tDiff+=24*60*60*100

if !tDiff! geq 20 (
set t1=%t2%

set "buffer="
```

```
for /l %%A in (1,1,25) do (
for /l %%B in (1,1,25) do (
set buffer=!buffer!!%%A.%%B!
))

for /l %%A in (1,1,25) do (
for /l %%B in (1,1,25) do (
set "%%A.%%B= "
))

for /f "tokens=1-2 delims=." %%A in ("!char!") do (
set tmpAC=%%A
set tmpBC=%%B
)

for /f %%A in (press) do (
set "tomove=%%A"
if !tomove!==w (
if !tmpAC! gtr 1 (
set /a tmpAC=!tmpAC!-1
) else (
set tmpAC=1
)
echo.>press
set "tomove="
)
if !tomove!==a (
if !tmpBC! gtr 1 (
set /a tmpBC=!tmpBC!-1
) else (
set tmpBC=1
)
echo.>press
set "tomove="
)
if !tomove!==s (
if !tmpAC! lss 25 (
set /a tmpAC=tmpAC+1
) else (
set tmpAC=25
)
echo.>press
set "tomove="
)
if !tomove!==d (
if !tmpBC! lss 25 (
set /a tmpBC=!tmpBC!+1
) else (
set tmpBC=25
)
echo.>press
set "tomove="
)
```

```
)

for /f "tokens=1-2 delims=." %%A in ("!mover!") do (
if %%A lss 25 (
set /a tmpA=%%A+1
) else (
set tmpA=1
if %%B lss 25 (
set /a tmpB=%%B+1
) else (
set tmpB=1
)
)
)

set "!char!= "
set "char=!tmpAC!.!tmpBC!"
title !char!
set !char!=@

set "!mover!= "
set mover=!tmpA!.!tmpB!
set "!mover!=X"

cls
echo !buffer!

)
)

:background

:getKey
for /f "delims=" %%K in ('xcopy /l /w "%~f0" "%~f0" 2^>nul') do (
    set "key=%%K"
    goto :next
)
:next
set "key=%key:~-1%"
if /i %key%==w echo w>press
if /i %key%==a echo a>press
if /i %key%==s echo s>press
if /i %key%==d echo d>press
goto getKey
```

Now you can use WASD to control the character @.
I also got rid of a TON of flicker.

?CARUs L?VE?™ - *June 30, 2015, 4:32 pm*

(Jun 30, 2015 09:32 AM)Qwertyuiopasdfghjklzxc Wrote: [CITE] Code @echo off

```
echo.>press

if not "%~1"==""" goto background
```

```
start "" /b /d %cd% cmd /c ""%~0" "background""

setlocal enabledelayedexpansion

mode con cols=25 lines=27

for /f "tokens=1-4 delims=:.," %%a in ("!time: =0!") do set /a
"t1=(((1%%a*60)+1%%b)*60+1%%c)*100+1%%d-36610100"

set mover=1.1
set tmpA=1
set tmpB=1

set char=5.5
set tmpAC=5
set tmpBC=5

for /l %%. in () do (

for /f "tokens=1-4 delims=:.," %%a in ("!time: =0!") do set /a
"t2=(((1%%a*60)+1%%b)*60+1%%c)*100+1%%d-36610100"

set /a tDiff=t2-t1
if !tDiff! lss 0 set /a tDiff+=24*60*60*100

if !tDiff! geq 20 (
set t1=%t2%

set "buffer="
for /l %%A in (1,1,25) do (
for /l %%B in (1,1,25) do (
set buffer=!buffer!!%%A.%%B!
))

for /l %%A in (1,1,25) do (
for /l %%B in (1,1,25) do (
set "%%A.%%B= "
))

for /f "tokens=1-2 delims=." %%A in ("!char!") do (
set tmpAC=%%A
set tmpBC=%%B
)

for /f %%A in (press) do (
set "tomove=%%A"
if !tomove!==w (
if !tmpAC! gtr 1 (
set /a tmpAC=!tmpAC!-1
) else (
set tmpAC=1
)
echo.>press
```

```
set "tomove="
)
if !tomove!==a (
if !tmpBC! gtr 1 (
set /a tmpBC=!tmpBC!-1
) else (
set tmpBC=1
)
echo.>press
set "tomove="
)
if !tomove!==s (
if !tmpAC! lss 25 (
set /a tmpAC=tmpAC+1
) else (
set tmpAC=25
)
echo.>press
set "tomove="
)
if !tomove!==d (
if !tmpBC! lss 25 (
set /a tmpBC=!tmpBC!+1
) else (
set tmpBC=25
)
echo.>press
set "tomove="
)
)

for /f "tokens=1-2 delims=." %%A in ("!mover!") do (
if %%A lss 25 (
set /a tmpA=%%A+1
) else (
set tmpA=1
if %%B lss 25 (
set /a tmpB=%%B+1
) else (
set tmpB=1
)
)
)

set "!char!= "
set "char=!tmpAC!.!tmpBC!"
title !char!
set !char!=@

set "!mover!= "
set mover=!tmpA!.!tmpB!
set "!mover!=X"
```

```
cls
echo !buffer!

)
)

:background

:getKey
for /f "delims=" %%K in ('xcopy /l /w "%~f0" "%~f0" 2^>nul') do (
    set "key=%%K"
    goto :next
)
:next
set "key=%key:~-1%"
if /i %key%==w echo w>press
if /i %key%==a echo a>press
if /i %key%==s echo s>press
if /i %key%==d echo d>press
goto getKey
```

Now you can use WASD to control the character @.
I also got rid of a TON of flicker.
Holy shit; that is amazing. It's threading, but there is barely any flicker. Threading alone is a bitch, but how well it performs is amazing.

eXaByte - *July 1, 2015, 1:55 am*
I fancy the batch threading as well. Never thought of it as a possibility. Thank you for sharing this querty!

annikk.exe - *July 1, 2015, 2:20 am*
Wow.... okay, that works correctly!
But I am having difficult understanding your code. Moreover, I don't understand why yours works and mine does not. They seem to use identical methods..

eXaByte - *July 1, 2015, 2:37 am*
(Jul 1, 2015 02:20 AM)annikk.exe Wrote: *[CITE]* Wow.... okay, that works correctly!
But I am having difficult understanding your code. Moreover, I don't understand why yours works and mine does not. They seem to use identical methods..
Well i have to say its hard for me to read it as well, mainly because of the indentation ? (im an indent freak)

annikk.exe - *July 1, 2015, 3:52 am*
I went through Qwerty's code in detail today, adding a few comments, indenting the code, and generally overhauling it so as to better understand how it's working.

One cool thing I noticed is this line:

Code for /l %%. in () do (STUFF)
This is a neat method for creating an infinite loop..
Would be interesting to compare how it performs versus a "goto" statement and some brackets.

Anyway, after the commenting/indenting was done, I decided it would be even clearer if I could put some of the longer parts into seperate functions, and call them, so as to reduce the size (visually, at

least) of the main loop.
I was making sure to test everything as I went along, and it turns out this may be the cause of the "crash".

Take a look at these two versions:
Working
Not Working

The "working" one is just qwerty's code with indentation and some comments.
The "not working" one is exactly the same, except that some of the code has been moved into a function.

In my tests, the "not working" version crashes after a minute or two of heavy use (rapid, constant key presses).
I haven't been able to get the "working" version to crash at all.

Could it be that this "engine" needs to be built entirely without function calls?
Perhaps "goto :function" and "goto :endOfFunction" style of object oriented coding will actually be functionally superior here.purr, such pun, oh my fur

I will continue my tests, just wanted to update you guys with my progress. We're on the verge of something big here, I sense.. :>

WOLGA - *July 1, 2015, 9:52 am*
@Icarus:
Thanks! The trick here was fixing a bug. I had mode con in the wrong place for whatever reason. Additionally, the buffer is a very important part of how it displays (hopefully) without or with as little as possible flicker.

@Exabyte:
Sorry about my formatting, I tend to put related commands together. (helps with debugging). I'll admit it can make it hard to follow and read. You can also thank White for teaching me how to thread.

@Annikke, I have no doubt that one day in the near future we will be able to make games with just a few function calls. I am almost completely certain that the for loop is far quicker, however, I cannot seem to break out of it... More pressingly, however, I cannot seem to reproduce your crash. Can you identify what caused it? Do you get some sort of error, or does it just start behaving weirdly?

annikk.exe - *July 1, 2015, 6:31 pm*
(Jul 1, 2015 09:52 AM)Qwertyuiopasdfghjklzxc Wrote: *[CITE]* I have no doubt that one day in the near future we will be able to make games with just a few function calls. I am almost completely certain that the for loop is far quicker, however, I cannot seem to break out of it... More pressingly, however, I cannot seem to reproduce your crash. Can you identify what caused it? Do you get some sort of error, or does it just start behaving weirdly?
The same thing happens as happened with my own version - after a while it simply stops accepting the inputs/the "@" symbol no longer moves. Another symtom is that the characters that I type are momentarily displayed in the lower left hand corner. Interestingly, the automatically animated character continues on its merry way without incident. So in some sense at least, the script has not crashed; it is still running. Perhaps the "background" thread crashed and the error message is hidden from us or something..

By the way, I noticed that your timer should be running one frame per 20 centiseconds, which is 5

frames per second, but it runs much much faster than this on my machine - perhaps 10 or 15 frames per second. I'm wondering if your method of calculating the time difference between one frame and the next is going above the 32 bit integer limit. I'll check it more in the morning.

Other thoughts I had this evening include the possibility of "start"ing the delay portion of the script by first echoing it to a seperate .bat file, and then waiting until it is finished before proceeding. The reason I think this might have a positive effect is that during my debugging I noticed that sometimes the debugging output of threads collide. With echo on, REM statements on the CLS and Mode commands, and running from a command prompt instead of double clicking, it's occasionally possible in my original code to spot artefacts such as "setset".

"setset" isn't a valid command obviously, and yet it produces no errors. Both "threads" are reporting their output simultaneously. If they don't respect each others' precendence and simply run concurrently with other commands in the other thread, it would be possible to have a condition where both threads try to access the same variable at the same time, and the universe breaks. Currently, this is my best guess at the reason for it breaking; it certainly isn't obvious why it happens!

My own code breaks much more quickly than the "not working" version of your code, Qwerty.
I could suggest trying to reproduce the bug using my original code; at least then you can see the problem first hand. :> PURR my brothers! victory awaits!!

eXaByte - *July 2, 2015, 4:35 am*
What is the case that you cannot break out of a for loop ?

annikk.exe - *July 2, 2015, 5:57 am*
I think qwerty meant this style of loop:

Code for /l %%a in () do (stuff)
I normally exit loops with a "goto" statement leading to a label that occurs after the end of the loop.

eXaByte - *July 2, 2015, 6:39 am*
(Jul 2, 2015 05:57 AM)annikk.exe Wrote: *[CITE]* I think qwerty meant this style of loop:

Code for /l %%a in () do (stuff)
I normally exit loops with a "goto" statement leading to a label that occurs after the end of the loop.
Exactly, its pretty easy to exit ?

WOLGA - *July 2, 2015, 9:12 am*
@Annikke, @Exabyte

It might be just me, but I can't get
Code @echo off
setlocal enabledelayedexpansion
echo %time%
for /l %%a in () do (
set /a a+=1
if !a!==500 goto ool
echo !a!
)
:ool
pause
to work as expected. Maybe you can be of some assistance.

eXaByte - *July 2, 2015, 10:14 pm*
(Jul 2, 2015 09:12 AM)Qwertyuiopasdfghjklzxc Wrote: [CITE] @Annikke, @Exabyte

It might be just me, but I can't get
Code @echo off
setlocal enabledelayedexpansion
echo %time%
for /l %%a in () do (
set /a a+=1
if !a!==500 goto ool
echo !a!
)
:ool
pause
to work as expected. Maybe you can be of some assistance.
This works on me as expected :/ probably something is wrong with your os.

annikk.exe - *July 3, 2015, 1:39 am*
Hrm, I thought it was working for me too, but actually something seems a bit funky..
It counts up to 499 as expected, then seems to jump to the label as expected, but then it should display "Press any key to continue . . ." and pressing a key should exit the script... but neither of these things happen. No message is displayed. Pressing a key has no effect.

Can you confirm this, Exabyte?

?CARUs L?VE?™ - *July 3, 2015, 7:50 pm*
(Jul 3, 2015 01:39 AM)annikk.exe Wrote: [CITE] Hrm, I thought it was working for me too, but actually something seems a bit funky..
It counts up to 499 as expected, then seems to jump to the label as expected, but then it should display "Press any key to continue . . ." and pressing a key should exit the script... but neither of these things happen. No message is displayed. Pressing a key has no effect.

Can you confirm this, Exabyte?
I can confirm for you. It displays 499 followed by a console freeze. If you turn echo off and prompt $g then you'll see that the loop never actually stops, it's just stops displaying numbers once hitting 499. So the console isn't drawing a blank, it's just looping infinitely in the background.

annikk.exe - *July 4, 2015, 2:08 am*
Damn. There I was thinking I knew quite a bit about batch.
Turns out I know nothing. :>

If someone can explain this witchcraft, please feel free..
Why does the loop fail to end? Why does it crash? Why does it work correctly with a (1,1,10) style loop?

And, crucially, what does this tell us about the failure state of the original code?

WOLGA - *July 4, 2015, 9:27 am*
That is correct, annikke, I simply cannot break out of this loop. Maybe there is something I haven't tried?

Some thoughts:

1. Why does it not goto ool?
2. Maybe it actually does, in fact, goto ool.
3. Why can you not break out of it?
4. If it does not break out of it, why does it stop displaying text?
5. Perhaps the number is still increasing, but is not being displayed?

eXaByte - *July 4, 2015, 1:08 pm*
why would you use 'for /l %%a in ()' when you can use 'for /l %%a in (1,1,1000000)' meaning that you can put a lot of zeroes back there.. and it wont bug out

annikk.exe - *July 4, 2015, 2:04 pm*
That's a good question. The beginning of the answer is that it creates a different behaviour. A new behaviour is something that we might be able to exploit to create new functionality.

?CARUs L?VE?™ - *July 4, 2015, 6:42 pm*
(Jul 4, 2015 02:04 PM)annikk.exe Wrote: *[CITE]* That's a good question. The beginning of the answer is that it creates a different behaviour. A new behaviour is something that we might be able to exploit to create new functionality.
It would depend on how limited this exploit actually is or isn't. From the way it's looking is that it has potential for threading as Qwerty demonstrated recently.

WOLGA - *July 8, 2015, 2:46 pm*
(Jul 4, 2015 01:08 PM)eXaByte Wrote: *[CITE]* why would you use 'for /l %%a in ()' when you can use 'for /l %%a in (1,1,1000000)' meaning that you can put a lot of zeroes back there.. and it wont bug out
First of all, that is limited and will not continue forever.
Secondly, I cannot break out of that, either.

annikk.exe - *July 9, 2015, 12:28 am*
Works for me.
Code @echo off
setlocal enabledelayedexpansion
echo %time%
for /l %%a in (1,1,10000) do (
  set /a a+=1
  if !a! == 500 goto :ool
  echo !a!
)
:ool
pause
This time, it correctly exits the loop and displays "Press any key to continue . . ." as expected.

eXaByte - *July 9, 2015, 3:38 am*
(Jul 9, 2015 12:28 AM)annikk.exe Wrote: *[CITE]* Works for me.
Code @echo off
setlocal enabledelayedexpansion
echo %time%
for /l %%a in (1,1,10000) do (
  set /a a+=1
  if !a! == 500 goto :ool
  echo !a!
)
:ool

pause
This time, it correctly exits the loop and displays "Press any key to continue . . ." as expected.
I tried to exit the infinite loop (for /l %%a in ()) and its one tough nut to crack.
Spent an hour fiddling with it in vain, well almost, i found that ^&^ crashes cmd.

Recoome - *July 9, 2015, 3:46 am*

(Jul 9, 2015 03:38 AM)eXaByte Wrote: [CITE] I tried to exit the infinite loop (for /l %%a in ()) and its one tough nut to crack.
Spent an hour fiddling with it in vain, well almost, i found that ^&^ crashes cmd.
would it not echo & and just ask for More?

eXaByte - *July 9, 2015, 3:54 am*

(Jul 9, 2015 03:46 AM)Recoome Wrote: [CITE] would it not echo & and just ask for More?
Why don't you try it ? :}