



# **Batch files** for the Windows Operating System:

---

## **Lecture 9**

Mr. Uttam Acharya



# Learning Outcomes

---

## **By the end of this lecture you will:**

- How to create a batch file (Windows OS) and execute it
- Learnt about some basic commands to get you started (programming)
  - **Setup** commands
  - **Control** commands
- Some advance batch file examples

# Creating a Batch Files (revisited)

- A **batch file** is a script file in that can be run under Microsoft Windows.
- It consists of a series of commands executed by the command-line interpreter
- It is stored in a plain text file with a **.bat** extension
  - Creating a batch file = **c:\notepad.exe** **test.bat**

Application used to  
construct program

Name of executable  
program

```
CA: Command Prompt
L:\4cs015>dir
Volume in drive L is ADATA UFD
Volume Serial Number is 4C47-63BA

Directory of L:\4cs015

22/10/2019  09:15    <DIR>        .
22/10/2019  09:15    <DIR>        ..
22/10/2019  10:53    <DIR>        old
22/10/2019  10:22    <DIR>        live
28/10/2019  09:34             0 test.txt
28/10/2019  09:37             0 test.bat
                2 File(s)              0 bytes
                4 Dir(s)  7,019,978,752 bytes free

L:\4cs015>
```

# Batch File Commands (setup)

---

- Program setup Example
  - **TITLE**- Edit the title of the window
  - **REM** - Inserts a comment line in the program
  - **ECHO** - Displays the text on the monitor
  - **@ECHO OFF** - Hides the text that is normally displayed
    - '@' symbol will prevent the 'echo off' command from being seen on the screen
  - Run

# Batch File Commands (control)

---

- Program control
  - **MKDIR** – Creates a directory
  - **COPY** - Copy a file or files (XCOPY – extended version)
  - **FOR/DO** - This command lets you specify actions with loops
  - **START** - Run a file with its default application
  - **IF/THEN/ELSE** – This command sequence allows for selective branching
  - [recallMe Example Run](#)

# Overview Example: Simple batch file execution

The image shows a Notepad window titled 'test - Notepad' with a menu bar (File, Edit, Format, View, Help) and a text area containing a batch file script. A teal box labeled 'Test.bat' is positioned above the script. A smaller command prompt window titled 'my first bat file' is overlaid on the Notepad window, showing the execution of the batch file. Arrows indicate the flow of execution: a red arrow points from the first line of the batch file to the command prompt title bar; a purple arrow points from the second line to the command prompt prompt; a green arrow points from the third line to the command prompt output; a yellow arrow points from the fourth line to the command prompt output; and a blue arrow points from the fifth line to the command prompt output.

```
title my first bat file
echo
rem !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
@echo off

rem ###start an application#####
start winword.exe

rem ###create a directory from current location###
mkdir test

rem ### copy a file in current location####
copy test.bat test2.bat

REM ###I will use the current location of the batchfile for file copy .\ ###
REM ###This moves any files with a .txt extension from source to target ###
REM ###%%f is a variable in batchfile - %f used with commandline - case sensitive###
rem ###the /Y switch is used to turn off commandline prompting #####
FOR %%f IN (*.txt) DO COPY .\"%%f\" .\text /Y
```

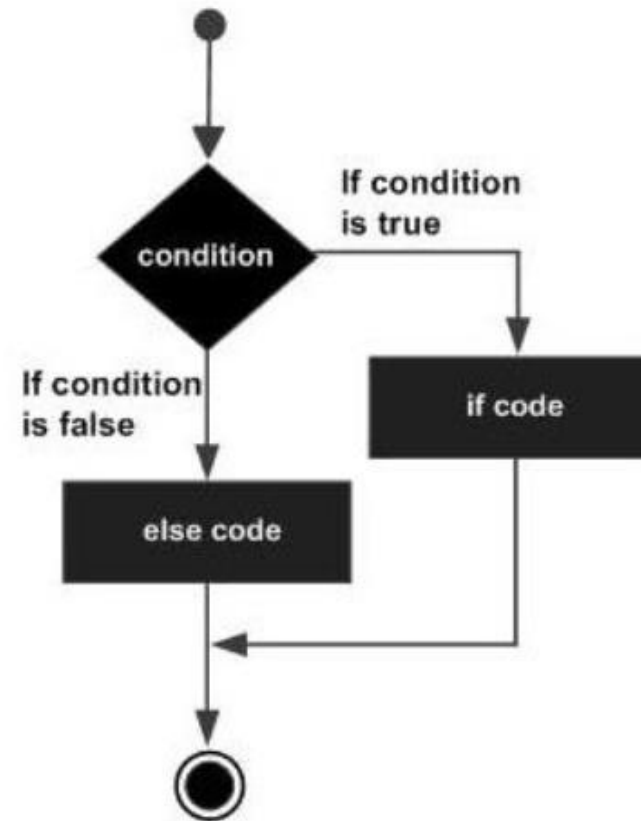
my first bat file

```
C:\Users\in7141>rem !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
A subdirectory or file test already exists.
1 file(s) copied.
1 file(s) copied.
C:\Users\in7141>
```

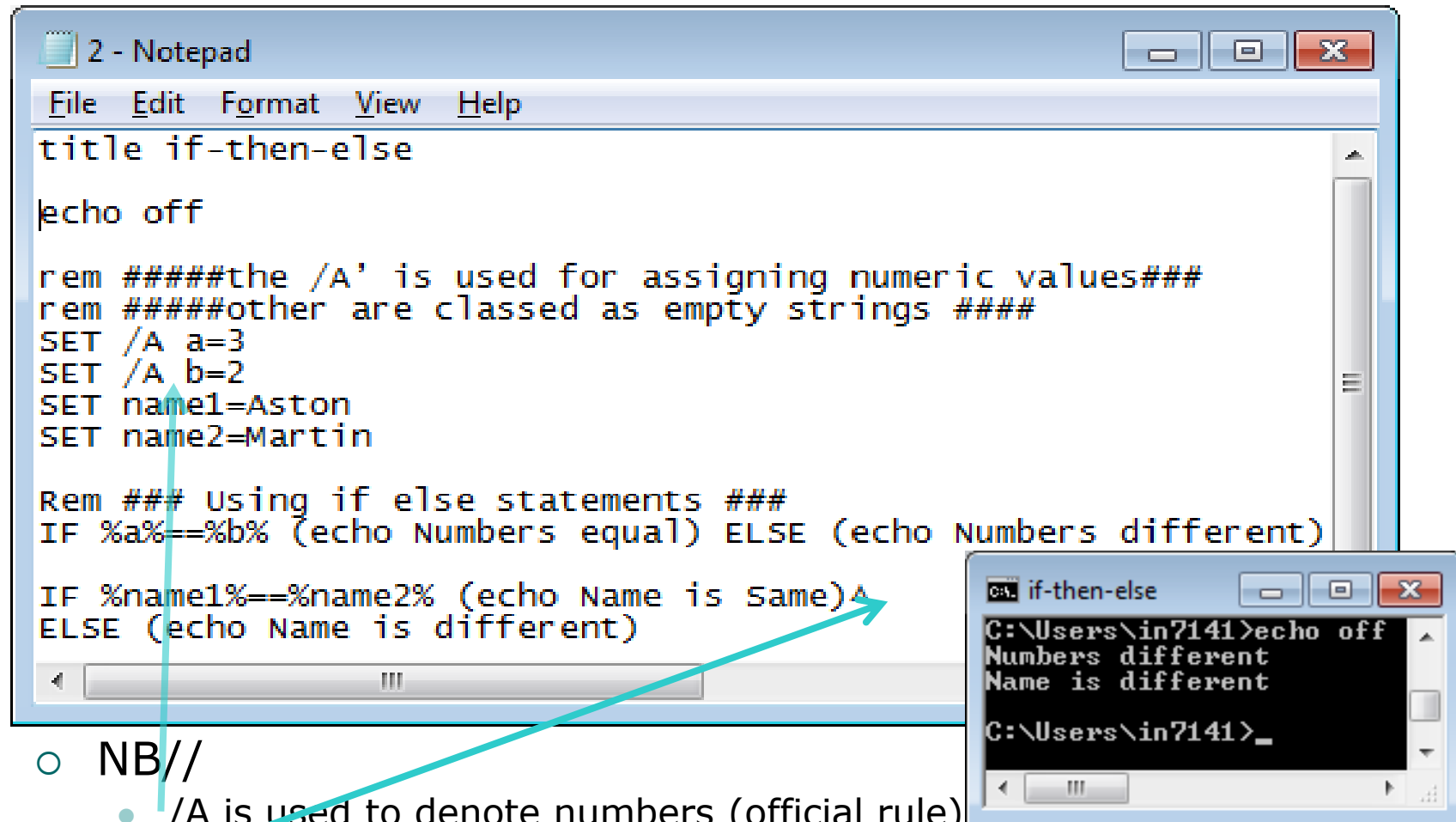
Exmple

# Batchfile: If-Then-Else

- The following is the general form of this statement -
- **IF** (condition)
- **THEN** (do something)
- **ELSE** (do something else)



# Example: If-Then-Else



The image shows two windows. The main window is a Notepad editor titled '2 - Notepad' with a menu bar (File, Edit, Format, View, Help). It contains a batch script with the following content:

```
title if-then-else

echo off

rem #####the /A' is used for assigning numeric values###
rem #####other are classed as empty strings #####
SET /A a=3
SET /A b=2
SET name1=Aston
SET name2=Martin

Rem ### Using if else statements ###
IF %a%==%b% (echo Numbers equal) ELSE (echo Numbers different)

IF %name1%==%name2% (echo Name is Same)^
ELSE (echo Name is different)
```

The second window is a Command Prompt titled 'if-then-else' showing the execution of the script. The output is:

```
C:\Users\in7141>echo off
Numbers different
Name is different
C:\Users\in7141>_
```

Two red arrows point from the text 'NB//' to the script. One arrow points to the line 'SET /A a=3' and the other points to the line 'IF %name1%==%name2% (echo Name is Same)^'.

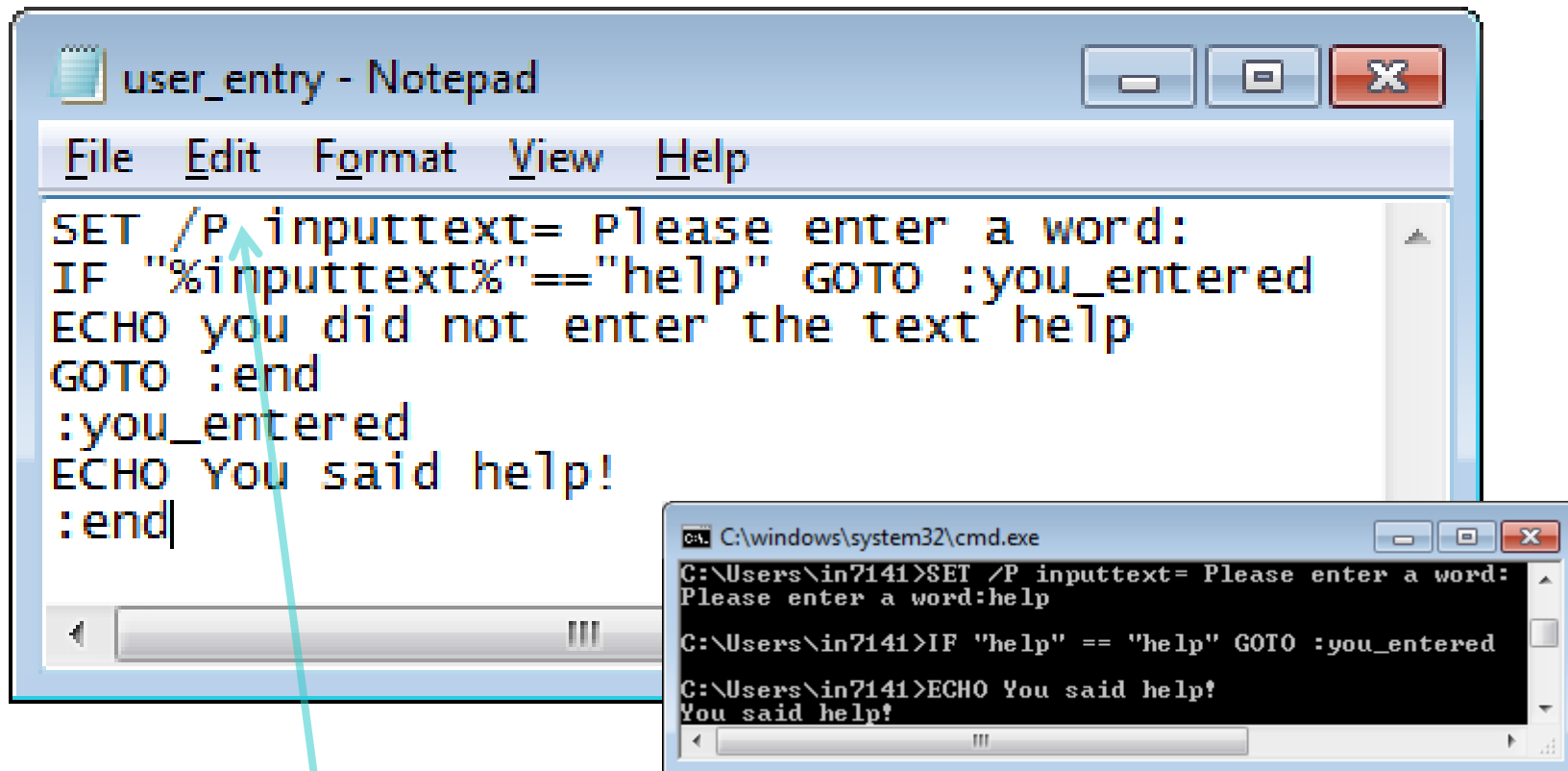
○ NB//

- /A is used to denote numbers (official rule)
- ^ (carrot) is used for continuing a command-line on the next line

[example](#) [Run](#)



# User Entry



The image shows two windows. The top window is titled 'user\_entry - Notepad' and contains the following batch script:

```
SET /P inputtext= Please enter a word:
IF "%inputtext%"=="help" GOTO :you_entered
ECHO you did not enter the text help
GOTO :end
:you_entered
ECHO You said help!
:end
```

The bottom window is a Command Prompt titled 'C:\windows\system32\cmd.exe'. It shows the execution of the batch script:

```
C:\Users\in7141>SET /P inputtext= Please enter a word:
Please enter a word:help

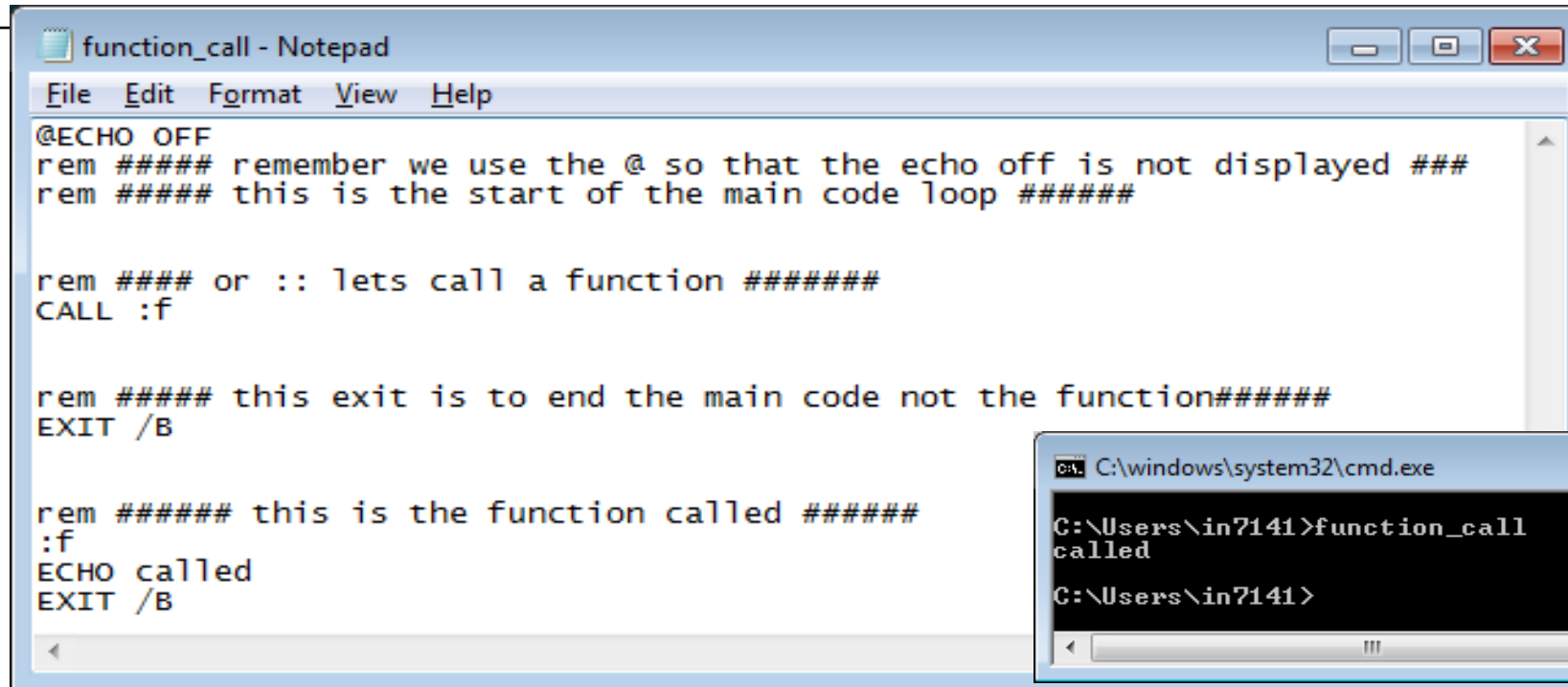
C:\Users\in7141>IF "help" == "help" GOTO :you_entered

C:\Users\in7141>ECHO You said help!
You said help!
```

A blue arrow points from the `SET /P` command in the Notepad window to the Command Prompt window, indicating the execution of that command.

- NB// The `'/P'` switch tells the command interpreter to prompt the user for an input which is saved into the variable which is stored as an environment variable which can be used later in the code. [Example run](#)

# Function Call(s)



The screenshot shows a Notepad window titled 'function\_call - Notepad' containing a batch script. The script uses the `@ECHO OFF` command to suppress command echoing. It includes several `rem` (comment) lines explaining the script's structure. The main code loop starts with `CALL :f`. An `EXIT /B` command is used to end the main code. A function named `:f` is defined, which echoes the word 'called' and then exits the function with `EXIT /B`. A Command Prompt window is overlaid on the bottom right, showing the execution of the batch file. The prompt `C:\Users\in7141>function_call` is entered, and the output `called` is displayed, followed by the prompt `C:\Users\in7141>`.

```
function_call - Notepad
File Edit Format View Help
@ECHO OFF
rem ##### remember we use the @ so that the echo off is not displayed ###
rem ##### this is the start of the main code loop #####

rem ##### or :: lets call a function #####
CALL :f

rem ##### this exit is to end the main code not the function#####
EXIT /B

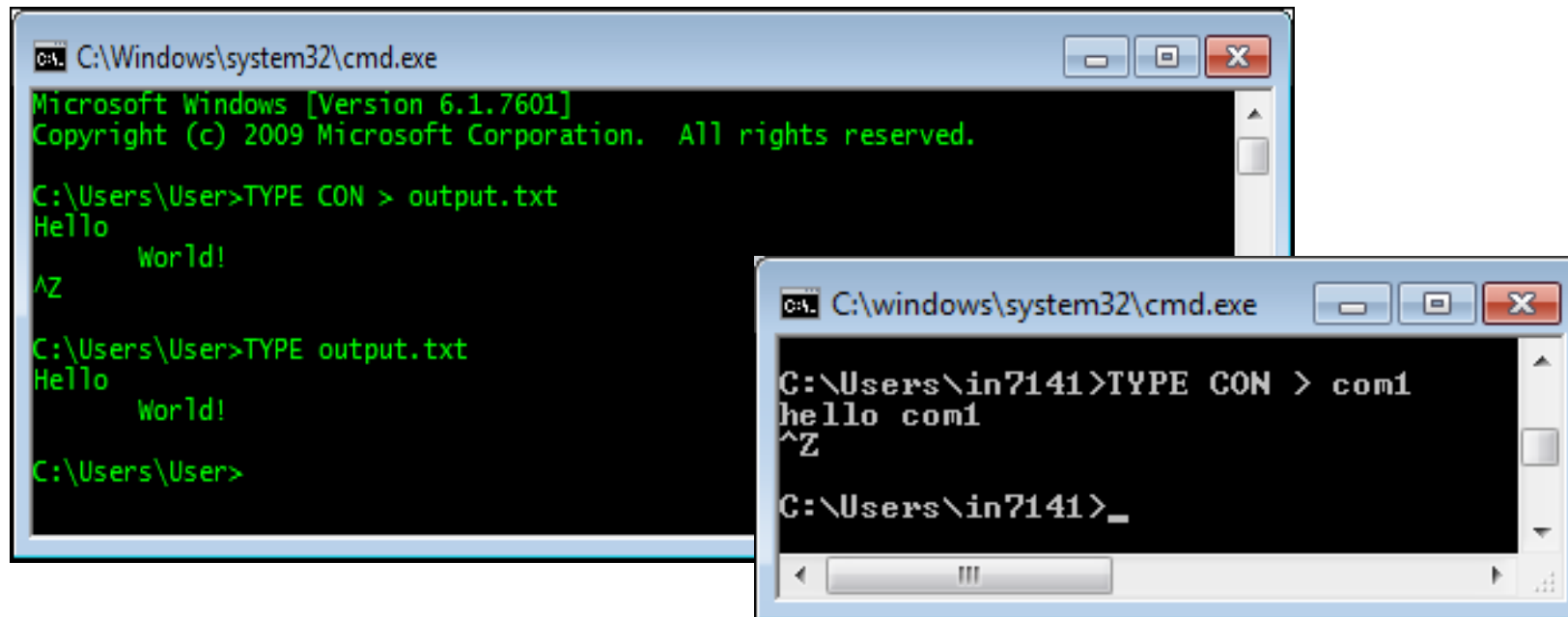
rem ##### this is the function called #####
:f
ECHO called
EXIT /B

C:\windows\system32\cmd.exe
C:\Users\in7141>function_call
called
C:\Users\in7141>
```

- We use the **CALL** keyword to invoke the function and pass any arguments to the function [Example Run](#)
- The **EXIT /B** will stop execution of a batch file or subroutine and return control to the command processor.

# Re-directing Input / Output

- You can direct commandline input by simply redirecting the command prompt's own **stdin**, called **CON**
- This can be directed to -
  - A file (in example 'output.txt')
  - Communication port (com1, com2, LPT1, etc.)
  - Using CTRL+Z, which sends the end-of-file (EOF) character to stop.



The image shows two overlapping Windows command prompt windows. The background window is titled 'C:\Windows\system32\cmd.exe' and shows the following text: 'Microsoft Windows [Version 6.1.7601] Copyright (c) 2009 Microsoft Corporation. All rights reserved. C:\Users\User>TYPE CON > output.txt Hello World! ^Z C:\Users\User>TYPE output.txt Hello World! C:\Users\User>'. The foreground window is titled 'C:\windows\system32\cmd.exe' and shows: 'C:\Users\in7141>TYPE CON > com1 hello com1 ^Z C:\Users\in7141>\_'. Both windows have standard Windows window controls (minimize, maximize, close) in the top right corner.

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\User>TYPE CON > output.txt
Hello
World!
^Z

C:\Users\User>TYPE output.txt
Hello
World!

C:\Users\User>
```

```
C:\windows\system32\cmd.exe
C:\Users\in7141>TYPE CON > com1
hello com1
^Z

C:\Users\in7141>_
```

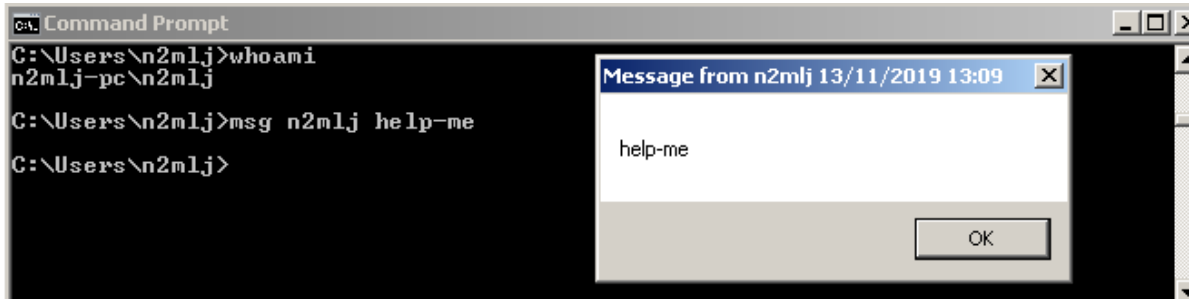
# Taster: More Advanced Control

---

- Swap (exchange) Mouse Button control (*only 2 lines of code!*)
- **@echo off**  
**Rundll32 User32, SwapMouseButton**
- To reverse the action on a University Computer simply reboot the machine
- For a Home machine –
  - Search "mouse" in the start menu & open it
  - The mouse dialog box will appear
  - In the buttons tab - Untick the option "switch primary and secondary buttons"

# Network BAT file

- **BAT (for example test.bat)**
- Using msg to send **network** messages to individual PC's
- @echo off
- msg n2mlj
- 
- n2mlj – computer name discovered by using **whoami** at command prompt
- *help-me* – text message sent



# Testing for IP addresses

---

- **Pinger – Testing IP network address example**

```
@echo off
title Pinger
set /p target=Enter IP address or URL:
ping %target%
pause
```

- **This outputs:**

```
Enter IP address or URL: google.com
Pinging google.com [216.58.220.206] with 32 bytes of data:
Reply from 216.58.220.206: bytes=32 time=26ms TTL=57
Reply from 216.58.220.206: bytes=32 time=25ms TTL=57
Reply from 216.58.220.206: bytes=32 time=27ms TTL=57
Reply from 216.58.220.206: bytes=32 time=25ms TTL=57
```

- *Pressing CTRL+C forcefully stops any running command*

# Turn-off Computer

---

- **Shutdown timer – This script closes down the windows operating system**
- The batch file schedules a shutdown user input where –
  - /p = user input,
  - /a = numerical value
  - CMDline shutdown is also used for log-off, reboot, etc.
  - Switches -s = shutdown, -t = time

```
@echo off
```

```
title Shutdown System
```

```
set /p min =Enter minutes to wait until shutdown:
```

```
set /a sec=%min%*60
```

```
shutdown -s -t %sec%
```

# Kill a task

---

- To kill an individual task (application or process)
- The task list can be view on your own computer by using CTRL+ALT+DEL
- The processes keep the system alive
- Many applications use the processes as services for them to execute

Code: *This will force termination of the firefox browser*

- @echo off  
taskkill /im chrome.exe /f
- pause

- Key

taskkill - use to terminate a process

/f - forceably termination of the process

im - this is the the image name associated with software (use task manager)

firefox.exe - the image name that you wanted to terminate.



# System Crash

- The **fork bomb** is the equivalent of a DDoS (distributed denial-of-service) attack on your operating system.
- It aims to deprive the system of memory (RAM), leaving nothing for other applications or the operating system's vital operations required to keep the systems running, hence crashing it.
- The fork bomb is not permanently harmful for a computer, just annoying.

The code:

```
:s  
start %0  
goto s
```

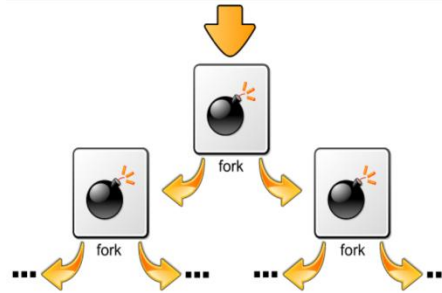
(1) Create a label **:s**

(2) Start means RUN – **%0** actually refers to the name of the batch file itself. So we are running the same file again.

(3) **goto s** – then forms the infinite loop

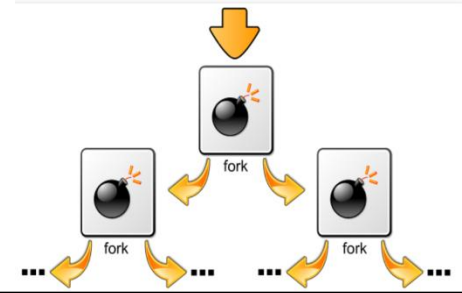
# System Crash (detail)

- Thus; every time the loop is completed another instance of the same program is started (both are then running and duplicate themselves, and so on.....)



- This doubling effect is a form of **exponential growth**.
- After one iteration of the loop, two programs ( $2^1$ ) are created. After another cycle, each of those two create another two for a total of four ( $2^2$ ). After 10 iterations we have 1024 ( $2^{10}$ ) instances of our little batch file. After 100 iterations we have  $2^{100} = \mathbf{1.267 \text{ nonillion}}$  ('nonillion' =  $10^{30}$  (US) or  $10^{54}$  (UK)).
- Many systems will not complete **50 iterations** before the system crashes.
- For such a simple script, each individual iteration would take **a few milliseconds**, thus it has a very quick effect on the computer.

# System Crash (protection)



- As with most unwanted or malicious script there is a way to protect you system.
- Any **antivirus** worth it's salt would be able to scan this suspicious executable file and warn the user before execution.
- As a fork bomb's mode of operation is entirely dependent on being able to **create new processes**
- One way of preventing a fork bomb s to limit the maximum number of processes that a system user can own (**not straight forward in Windows – 'MSconfig'**)
- On Linux, this can be achieved by using the *ulimit* utility; for example, the command **ulimit -u 30** would limit the affected user to a maximum of thirty owned processes.

For interest, the fork bomb in Linux - [https://en.wikipedia.org/wiki/Fork\\_bomb](https://en.wikipedia.org/wiki/Fork_bomb)

Bash

```
:(){ :|:& };;
```

# Rundll32.exe (run a DLL)

---

- The **rundll32.exe** process is responsible for running **DLLs** (dynamic link libraries) and placing them into memory
- It can be used for **malicious purposes** by an attacker allowing access to your computer from remote locations, stealing passwords, Internet banking and personal data.

*Code for opening a file with Windows' "Open as" dialog box*

**RUNDLL32 SHELL32.DLL,OpenAs\_RunDLL filename**

*Code for swapping your mouse button to left handed use (already shown – part 1)*

**RUNDLL32 USER32.DLL,SwapMouseButton**

*Code to open the device manager in windows*

**RUNDLL32 devmgr.dll DeviceManager\_Execute**

**NB//** Code to list all DLL's running use '**listdlls**' program -

<https://docs.microsoft.com/en-us/sysinternals/downloads/listdlls>

# Registry Entries (changes )

---

- You can directly make changes to software execution by changing associated software entries in the windows system register
- You use a program called '**reg edit**' to achieve this (\*\* WARNING this can make you O.S. completely un-useable)

- **Disable Mouse (using the windows register)**

```
@echo offset  
key="HKEY_LOCAL_MACHINE\system\CurrentControlSet\Service\Mouclass"  
reg delete %key%  
reg add %key% /v Start /t  
REG_DWORD /d 4
```

NB// To Rollback the changes:- Just Change the last line of above code with  
REG\_DWORD /d 3

# Auto-Launch at Windows Startup

There are two ways on how will you make your batch file runs at start up.

- (1) Creating a **registry key**
- (2) Copying the file to the **Startup** folder (*you could then hide the file so it is not visible*)
- C:\Users\uttambabu\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup

Method (1) - Note the location of file.bat (the file we

Copy our batch file to system32 folder

@echo off

copy file.bat "C:\windows\system32"

attrib +h "C:\windows\system32\file.bat"

reg add hklm\software\microsoft\windows\currentversion\run /v filedotbat /t

reg\_sz /d C:\windows\system32\file.bat /f

Change its attributes so it will be hidden (attrib/? = help)

Make a registry key for auto-startup

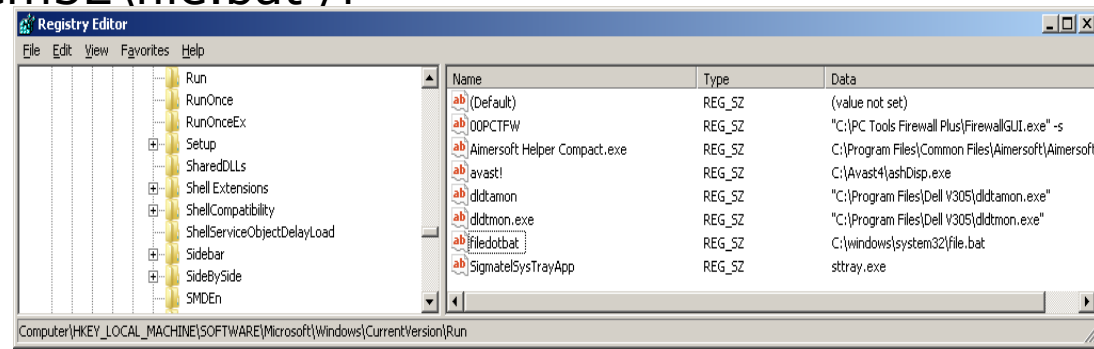
/v - use to specify a file name

/t - type of registry key

/d - the destination of the file to execute

/f - force to create a registry key

NB// filedotbat - name of the register entry  
(see Registry Editor Picture)



# Auto-Launch (method 2)

---

- This method for Windows is much easier.
- It works on the premiss that Windows always look's toward the startup folder when the system is re-starting

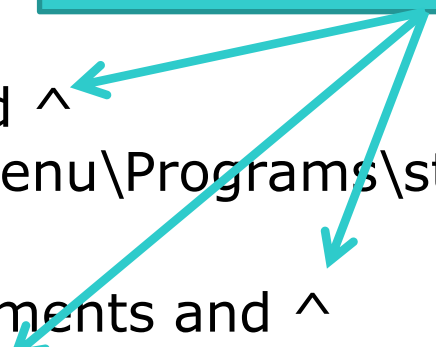
The code:

```
@echo off
```

```
copy file.bat "C:\Documents and  
Settings\%userprofile%\Start Menu\Programs\startup"
```

```
attrib +h copy file.bat "C:\Documents and  
Settings\%userprofile%\Start  
Menu\Programs\startup\file.bat"
```

NB// caret symbol ^ for  
continuing a single line





# Summary

---

- Add multiple command-line functions and actions into a sequence of events (creating a batch-file)
- You can use many simple commands -
  - dir, rmdir, mkdir, etc.
  - User entry - set /p
  - Loops for conditional actions - If/then/else
- We can also –
  - Execute **applications** with **switches** and **filenames**, from specific locations.
  - Call **DLL**'s (dynamic Link Library's) that can effect system behaviour





# Workshop

---

- Finish command line material for Windows and Linux and start batch-file programming
- Submission date for portfolio is last week of module – please see canvas for details.