Cygwin – command line windows

Get that Linux feeling - on Windows

http://cygwin.com/

Outline

1. What is Cygwin?
2. Why learn it?
3. The basic commands
4. Combining commands in scripts
5. How to get more information

What is Cygwin

Cygwin is:

1. a collection of tools which **provide a Linux look and feel environment** for Windows.
2. a DLL (cygwin1.dll) which acts as a Linux API layer **providing substantial Linux API functionality**.

Cygwin is not:

1. **a way to run native Linux apps on Windows**. You must rebuild your application *from source* if you want it to run on Windows.
2. **a way to magically make native Windows apps aware of UNIX®** functionality like signals, ptys, etc. Again, you need to build your apps *from source* if you want to take advantage of Cygwin functionality.

What is great about CYGWIN

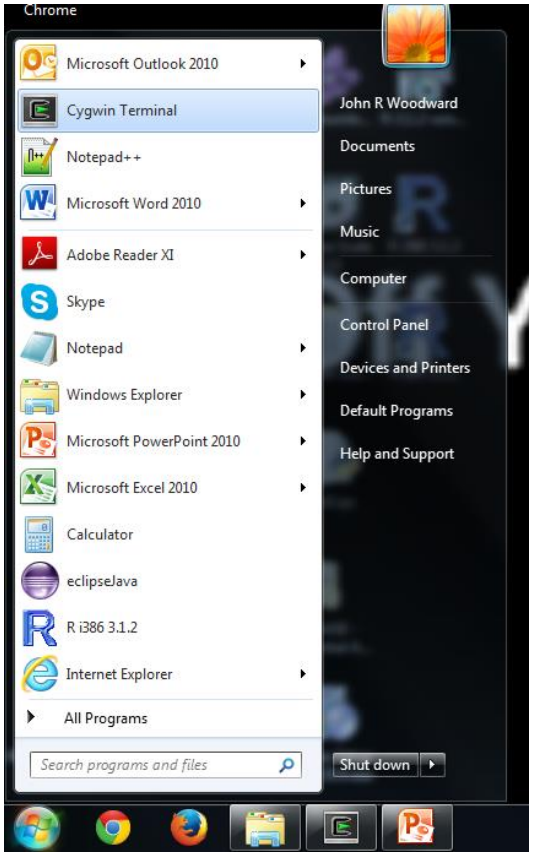
1. It is completely **free**!
2. **Low-barrier entry**, no-anxiety try out:
   -- Just download it and check it out. If you fail all you lost was 1-2 hours of time. But consider the possible payoff.
3. **Stable** environment (Unix-based) with lots of tools.
   -- All the **GNU compilers** and **tools** are available.
4. Easy to **install**.
5. No need to **dual-boot**. Great time saver!
6. **Shares files** with Windows.
7. Convenient **automatic** upgrade/update facilities

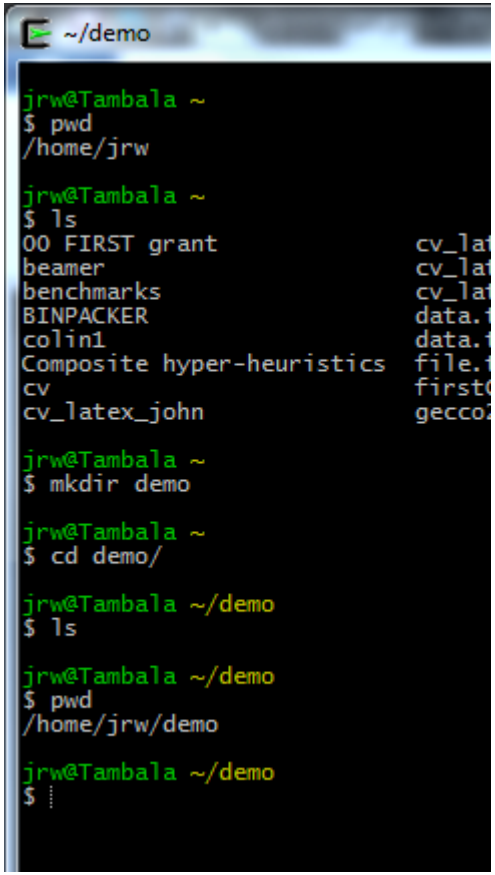
Starting Cygwin

1. Open your Cygwin Console by clicking:
2. **Start>All Programs>Cygwin>Cygwin Bash Shell**.
3. When you start a Cygwin Console, you are automatically sent to your Cygwin home directory - which usually corresponds to your Windows username. All Windows users on your system should have a home directory with a Windows path of:
4. **c:\cygwin\home\[Windows Username]**
5. To find out what your home directory name is type the '**pwd**' (i.e. **print working directory**) command in a newly opened Cygwin Console.

My home directory

- C:\cygwin\home\jrw





Why are command short

1. To list a directory why not use the command "**list**" – which lists all the files in the directory?
2. The command is "**ls**"
3. The reason is – in the early days of computing, there was not much memory (RAM), so commands were as short as possible. This convention has stuck.

Autocomplete – like predictive text

1. If you partly type a command and press “tab”, the computer will try to guess the rest of the command
2. e.g. pdfl “tab” will give pdflatex (on my system).
3. It will also try to complete filenames
4. e.g. pdflatex exa”tab” will give pdflatex example. (on my system).
5. Note also Cygwin is CASE SENSITIVE

Directory Structure – Duplicates Linux

Windows location	Cygwin Console
c:\cygwin\bin	/bin
c:\cygwin\etc	/etc
c:\cygwin\home	/home
c:\cygwin\home\administrator	/home/administrator
c:\cygwin\lib	/lib
c:\cygwin\tmp	/tmp
c:\cygwin\usr	/usr
c:\cygwin\var	/var
c:\cygwin	/

Cutting and Pasting Content from Windows to Cygwin

1. from the Windows application:
2. **highlight** the text to be copied;
3. **right-click** and select **copy** from right-click menu (or hit ctrl-c key combination);
4. go to Cygwin window:
5. right-click Cygwin **window header**
6. select **Edit>Paste** from the right-click menu
7. (can be different on different systems e.g. click mouse wheel)

Extracting Gzipped Tar files in Windows

1. The 7-Zip program can unzip and untar Linux gzipped tar files from Windows Explorer. Once installed, you simply right-click the file you want to unzip and/or untar in Windows Explorer, and select 7-Zip from the right-click menu. If your file is gzipped and tarred, you need to execute the 7-Zip extract twice: first on the .tgz file, and then again on the generated .tar file.
2. It is Open Source and licensed under LGPL. It can be obtained at:
3. http://www.7-zip.org/
4. You can also use the Cygwin tar command

Working with Files

cp <filename> <new filename>
copy - Make a copy of a file
cp -R <directory> <new directory>
Make a copy of a directory
mv <filename> <new filename>
move - Move or rename a file
rm <filename>
remove - Delete a file

File name expansion

ls *.* will list all files with a dot in the name

e.g. **CV.doc** but not **CV**

* Stands for zero or more characters.

ls *.??? Will list all files with a dot followed by three characters.

e.g. **Lecture.ppt** but not **Lecture.pptx**

? Stands for exactly one character.

ls ???*.* will list files with a filename at least three characters long (followed by a dot)

Very basic Regular Expressions

rm *[0-9].txt

Will remove all files with a digit before the ".txt"

rm *.[!t]*

Will remove all files which do not start with a "t" in the file extension.

ls -l| grep ^d will list only directories

ls -l| grep sh$ will list files ending with "sh"

Working with Directories

cd <directory> change directory - Change to the directory specified

ls List - Lists the files in the current directory
ls -l Lists the files and their attributes
mkdir <new directory name>
make directory - Create a new directory
pwd Path of working directory - tells you what directory you are in

cd .. Means move one level up in directory

rmdir <directory> removes the directory

Basic Commands 1

man	Displays the manual page for the given command (try "man man"). If you do not find the manual page for some of the commands given on these pages, I would guess that it is because it is one of the commands that are handled directly by bash - see if you find it at "man bash".
pwd	"Print Working Directory" - tells you in which directory you are.
ls	ls lists the files in the current directory
cat	(From "conCATenate" i believe). Reads the given file(s) and prints it to stdout. If not files are given it reads from stdin. Note that ^D (control-D) is "the unix end of file"
cd	"Change Directory". Give the name of the directory you want to be your new current directory - either relative to where you are, or absolute, relative to the file system tree "root" at "/".
cp	"Copy" a file: "cp hello1.c hello2.c" will make hello2.c a copy of hello1.c

Basic Commands 2

mv	move (and/or rename) a file.
rm	remove/delete a file.
mkdir	makes a new directory
rmdir	removes a directory
grep	"grep <pattern> [list of files]" will search through the files and print any lines that contains the pattern.
ps	Lists the processes that are running at the time.

Archiving/Extracting Files and Directories

tar -zcvf <filename> <directory> # create gzipped tar archive of <directory>

-z - filter the archive through gzip

-c - create a new archive

-v - verbosely list files processed

-f - use archive file

tar -xvzf <filename> # extract tarred, gzipped <filename> in current directory

-x - extract files from an archive

-v - verbosely list files processed

-z - filter the archive through gzip

-f - use archive file

Processes

notepad & (will run notepad in the background)

ps (will tell you what processes are running)

kill 1234 (will terminate process with ID 1234)

Control-Z (suspend a foreground process)

bg (puts a process into the background)

fg (puts a process into the forground)

YOU SHOULD execute command which are not "instant" (e.g. **ls**) with & e.g **notepad CV.txt &**

Pipes and redirection

<	"**program < file**" : "Take your input from the given file rather than the keyboard". For example "**cat < hello.c**" will type the contents of hello.c to the console. This is just like "**cat hello.c**" would do, of course, but in the first case cat would just read from stdin and write to stdout as usual while in the second case it would parse its command line to find a file name and open and read from it explicitly. **GOOD FOR TESTING A PROGRAM**
>	"**program > file**" : "Write the output from the program to the given file rather than to the console.". For example "**ls >dir.txt**" would make a directory listing and place it in dir.txt. **GOOD FOR SAVING OUTPUT**
|	"**program1 | program2**" : Let the output on program1's stdout be the input on program2's stdin. For example "**ls | wc** " would see if there was any files containing "test" in its name in the current directory. **GOOD FOR BUILDING COMPLEX COMMANDS**
``	("BackQuotes") **"program1 `program2`"** : The shell will, when preparing the command line for program1, run program2 and replace `program2` with the output from program2. **GOOD FOR EXECUTING PROGRAMS BEFORE HAND**

Using redirection for testing

- java prog.java
- (but you have to keep typing input)
- java prog.java < input.txt
- (all of the input is in input.txt)
- Even better
- java prog.java < input.txt > output.txt
- (all of the output is in output.txt)
- (you can do this with tools like junit)

Working with File Permissions:

1. **chmod u+x** <filename>
2. e.g. **chmod u+x myScript.sh**
   changes permission of the named file to executable. The extenstion “sh” mean script.
3. u - user, (this means you)
4. **+** - adds permissions
5. x - executable rights

Getting Help

1. You can get further details on any commands in Cygwin by using the '**info**' or the '**man**' command. These provide essentially the same information, but with slightly different formatting.
2. To obtain detailed information on the '**mv**' for example, you would type one of the following: **info mv** or **man mv** (to exit info or man, type the letter 'q' at the ":" prompt.)
3. (BETTER is just to google it – there are some very good tutorials – or see youtube)

Scripts as automation

- You can type commands at the command line and do everything you can do with a mouse (you can decide which way you like)
- The real bonus of command lines is if you have to do something repeatedly – you write a single script (program), then you do not have to keep writing the same commands

Example compiling java

- javac Hello.java
- java Hello.java
- You could write this as one script
- (call it “jh.sh”).
- Also you could write
- javac Hello.java && java Hello.java
- Why is this an improvements
- && is like “short circuit” in java

How to run a script - IMPORTANT

1. Make a file called “todo.sh” containing some commands
2. Make it executable with (change permissions)
3. **chmod 755 todo.sh**
4. Execute it with the following command
5. **./todo.sh** or **sh todo.sh**
6. (“.” is the current directory, alternatively you can add the directory to your path – not recommended – you can learn more)

dos2unix

1. Sometimes if you download a file off the web you might have issues with unix/windows
2. This can often be corrected with the command
3. dos2unix script.sh
4. If you do word count (wc script.sh) before and after – you will see that the file has changed length (number of characters)

Scripts - echo

echo hi

echo $1

echo bye

The above script

1. Prints “hi” on the screen
2. Prints the first argument eg. ./echo.sh xxxx will print xxxx (or “x y”)
3. Prints “bye”

Scripts first

echo hi

mkdir directory1

echo "I just made directory1“

The above script

Will print messages to the screen, and also make a directory called directory1

Script makeDirectory

```
for ((i=1; i <= 9; i++))
do
mkdir $i
done
```

The above script will make ten directories with the names 1, 2, 3, …, 9, 10

for/do/done repeats what is between do/done

Script read.sh

read x

echo $x

The first line of script read input from the user (as typed at the command line) and puts it into a variable called “x”

The second line prints “the value of the variable x” on the screen

Compare “echo x” prints x

To find out more

- Google “Cygwin/tutorial/beginners/guide/faq”
- http://www.voxforge.org/home/docs/cygwin-cheat-sheet
- http://ss64.com/bash/
- An alternative http://www.mingw.org/
- Lots of tools e.g. sed, awk, find, ….