Introduction to the Unix Shell

Ulrik Nyman

October 10th 2008

Last updated: October 9, 2008

Todays Agenda

- What is the shell?
- What is it good for?
- How does it work?

- Installing PuTTY and Xming
- Basic commands
- Exercises

What is the shell

- The shell was the first interactive way to use a computer
  - A huge step up from the batch systems
  - You might mistake the shell for a dinosaur, but...
- Is extremely flexible and easy to expand
  - Also a very effective and fast way to use a computer
  - Easier on the wrists than using a mouse

What is the shell

- The shell is kinda like a big Swiss Army knife
  - Only with extra springs, and small bombs in it
    - But it only does what you ask it to
  - It can do pretty much anything
  - No bells and whistles
- Note: Many shells exists, today we assume bash
- The terms “console” and “terminal” are also used instead of shell

Commands and Arguments

- Usually, a shell look like this:

- But may look slightly different

Commands and Arguments

- The shell does nothing by itself (including helping you)
- You must type in commands, like this

```
user@host:~> command argument1 argument2 ....
```

- The command decides what to do with arguments
- A command is usually a program (but not always)
- There are many commands
  - One does not need to know them all
  - Of 2617 I probably know 100

A few shortcuts

- You can use ↑ and ↓ to scroll through history
- Shortcuts:

Ctrl+C	Abort the current command
Ctrl+R	Search through command history
Ctrl+D	Exit shell (if enabled)
Ctrl+Z	Stopping current command
bg	Starting command in background again

Wildcards

- Wildcards are special characters used to match files
- Example: `ls *.pdf` will show all files ending with `.pdf`
- `*` means any character, any number of times
  - Example: `c*` all files that start with `c`
- There is full support for regular expressions
- But `*` will be enough for most operations

- Note: The expansion is done by the shell, NOT the program

Man pages

- Most commands have manuals
- These can be accessed through the `man` program
- Usage:

  ```
  man ls
  ```

  - Will show the manual for `ls`
- Also man pages for APIs, configuration files, etc.
- Do not be afraid of man pages

Combining Programs

- Unix contains many many commands
  - Each of them does very little
  - But can be combined in a very flexible way
- Unix philosophy:
  - Combining lots of small programs into one big
  - As opposed to just having one big program
  - This scheme has advantages and disadvantages
    - Flexibility $\asymp$ Ease of use

Combining Programs

- Pipes can combine programs by makings one the output of one program, the input to another program.
  - Like this: `command1 | command2`
  - This is very powerful
  - Its only text

Combining Programs

- Example:
  - There is no program to report the number of files in a directory
  - `ls` lists the files in a directory
  - `wc` counts the number of lines (among other things)
  - `ls|wc -l` shows the number of files in a directory
- Another example: `ls | grep pdf`
- Saving output to a file: `command > file`
- Input from a file: `command < file`

Listing files

- The `ls` command is the most used shell command
- It lists files
  - Directories are also files
- Without any arguments, `ls` will show the content of your working directory
  - `ls directory` will show the content of a directory
- Flags:

  -l Lists details about the files

  -a Lists “hidden” files (files starting with `.`)
- `ls -la` will show all files and detail about them

Moving Around

- The `cd` command changes your current working directory
- Entering a directory: `cd` *`directory`*
- Entering the above directory: `cd ..`
- `cd` without any arguments will take you to your home directory

- The command `pwd` prints your current working directory

Creating Files and Directories

- To create a directory: `mkdir` *`directory`*
  - Thats pretty much it...
- Usually files are created by higher level applications
  - To simply create a file, use: `touch` *`file`*
  - `touch` will also update the file modification time
- To see the content of a file: `cat` *`file`*
  - For larger files, use less: `less` *`file`*
  - or `more` *`file`* or `head` *`file`*
  - or `tail` *`file`*

Deleting Files

- The delete command is called rm (remove)
- Deleting a file: `rm` *`file`*
- Historically, `rm` could only delete files
  - To delete a directory, `rmdir` should be used
  - `rmdir` can only delete empty directories though
  - This became rather annoying, so `rm` was expanded
- To recursively delete, use `rm -r` *`directory`*
- Careful now, there is no undo in Unix

Copying & Moving

- The `cp file1 file2` copies `file1` to `file2`
  - Will overwrite `file2` if it exists!
  - To copy directories, the flag `-r` must be added
- Moving is the same: `mv file1 file2`
  - Again, this will overwrite `file2` if it exists
  - Question: The difference between a rename and a move is?
- Both `cp` and `mv` work with wildcards
  - But only when it makes sense
  - Example: `mv *.pdf docs/`
  - Not supported: `mv *.ps *.pdf`
  - Question: Why doesn't this work?

Identity

- The existential command: `whoami`
  - Which groups am I in: `groups`
  - Which groups are my neighbour in: `groups` *`user`*
- Who is logged in to the machine: `who`

Unix Permissions

- Unix permissions confuses most newcomers
- The permissions of a file looks like this:

```
drwxrws---  16 ulrik tav       4096 Sep 23  2004 visualS
-rw-rw-r--   1 ulrik ulrik 50492881 Sep 25 12:47 thesis.
```

- 9 permission flags:
  - read, write, execute × user, group, other
- Set read, write and execute for user: `chmod u+rwx`
- Changing group: `chgrp` *`group`* `file`
  - You can only change to a group which you are a member of
- Setting sticky bit `chmod g+s`

Unix Permissions

- Unix permissions confuses most newcomers
- The permissions of a file looks like this:

```
drwxrws---  16 ulrik tav       4096 Sep 23  2004 visualS
-rw-rw-r--   1 ulrik ulrik 50492881 Sep 25 12:47 thesis.
```

- 9 permission flags:
  - read, write, execute × user, group, other
- Set read, write and execute for user: `chmod u+rwx`
- Directory permissions

read	list
write	create / delete
execute	enter

Editors

- Emacs is not really considered a Unix tool
- `ed` is probably the “original” Unix editor
  - `ed` is really really strange and old
- `vi` later replaced ed
  - `vi` is very different from other editors
- There is no easy-to-learn editor with wide availability
- However `nano` is usually available.
  - If `nano` is not available, try `pico` (similar)
- In the bottom of `nano`/`pico` there is a small help screen
- Note: `^X` means Ctrl+X

Process Management

- List your current running processes: `ps`
  - List all processes: `ps -e`
- Dynamic listing: `top`
  - Can view per user, sort by CPU time, memory, etc.
- Kill a process (gracefully): `kill` *`PID`*
  - Kill a process (brutally): `kill -9` *`PID`*
  - Killall: `killall` *`processname`*
- Starting a process with low priority: `nice -n 19` *`command`*
  - Use this for long running tasks / experiments

Screen

- How to avoid killing a program when logging out?
  - Use screen: `screen`
    - Gives you a new shell to start your program
  - Detach screen: `Ctrl+A, Ctrl+D`
  - Reattach screen: `screen -r`
  - Again, use this for long running tasks / experiments

SSH and SCP

- The only way to access the network besides VPN
- Logging on to a remote machine: `ssh homer.cs.aau.dk`
  - With X forwarding: `ssh -Y machinename`
- Using SCP for file copying:
  - `scp homer.cs.aau.dk:/file localfile`
  - `scp localfile homer.cs.aau.dk:/file`
  - Also works with directories (add `-r`) and wildcards

Summary

- The shell can do everything
- But it is very different from graphical interfaces
  - The shell does not present options as the graphical interfaces
- You can start with a few commands and learn as you go
- The shell is a programming environment
- It takes time to become good with the shell
  - But one can become very effective with it

Exercises 1: Basic File Management

- List the files in you home directory
- List the hidden files, and with size and owner
- Create a file - and delete it
- Create a directory - and delete it
- Output the content of a file to the shell
- Remove a directory + contents (one command)
- Copy a file
- Rename a file
- If your file structure is messy, rearrange it

Exercises 2: Identity

- Execute the command that writes your username
- Who is logged into the machine
- Which groups are you member of
- What about the person next to you

Exercises 3: Permissions

- Create a directory and change the permissions so only you can access it
- Create a directory and change the permissions so your group can write in that directory.
- Set the sticky bit for the group and get one of your group members to create a sub directory.

- Many new students change the permission of the home directory, so only they can access it.
  - Why is this a bad idea?

Exercises 4: Process Management

- List your processes running in the shell
- List all the processes on the machine
- View the processes of yourself (press “u” in top)
  - And another user
- Create a sleeping process in one terminal and kill it in another
  - Create a sleeping process with: `sleep 10m`
- Create a sleeping process with low priority
- Create a sleeping process in a screen, log out, log in, and re-attach the screen

Exercises 5: SSH and SCP

- Use `ssh` to log on to another application server
- Copy a file from one application server to another
  - Use a filename that is not in use!

Exercises 6: grep

- Use `grep` to search all users home directories for `.tex` files containing the word `new`
- Count the number of lines on which new occurs