Practical Programming Methodology
(CMPUT-201)

Michael Buro

Lecture 2

- Getting started
- UNIX file system

Getting Started

Two ways of accessing a UNIX computer:

1. Sitting in front of it and typing in a command window
2. Connecting to it from a remote machine using ssh ("secure shell")

   `ssh ug01.cs.ualberta.ca`

Both require you to provide a userid and password

Command Shell

- In interactive mode, command line interface (text window with keyboard attached to it) e.g. "xterm"
- Issue operating system or internal shell commands directly via keyboard input; e.g.
  - ls (list directory contents) `ls -lrt`
  - cd (change directory) `cd workdir`
  - mv (move/rename) `mv old-file new-file`
  - mkdir (create directory) `mkdir AS1`
  - cp (copy file or directory) `cp -r dir backup`
  - rm (remove file or directory) `rm -rf dir`
  - cat (display file) `cat text`
  - echo (display string) `echo hello`
  - exit (quit shell) `exit`
  - man (command info) `man man`

UNIX File System

Data is stored in file systems which are usually located on harddisks

Persistent: data not lost when computer is switched off (unlike RAM)

- Hierarchical structure (tree)
- / represents the root directory
- Directories ("folders") can contain other directories and files (internal nodes)
- Files (leaves) are just sequences of bytes
- Files/directories are uniquely located by a directory path. E.g. `/home/user/AS1/foo.c`
- / is also used as directory separator

Shell continued

- Special directories:
  - / root directory, everything is stored beneath
  - . current directory `cp ./foo ./bar` = `cp foo bar`
  - .. parent directory `cd ../..` : 2 levels up
  - ~ home directory `cd ~/foo`
- Command history/editing
  - use arrow keys to navigate, <delete> or <backspace> keys to remove characters
- Simple programming language
  - variables, functions, command aliases
- Startup code in ~/.bashrc (when shell=bash)
  - customizations! `function ll() { ls -l "$@"; }`

Launching Programs

- Type program name (+ parameters) and hit the return key <ret>

  `ls<ret>`

  `emacs foo.c<ret>`
- Shell interprets the first word as command name and tries to locate a function definition with this name (see ~/.bashrc). If this fails it searches in the directories listed in variable $PATH (try `echo $PATH`)
- To detach program from terminal to run it in background type

  `command &<ret>` (= `command<ret><ctrl-z>bg<ret>`)

Wildcards

* matches all strings
? matches one character

Examples:

- `wc *.c`
  count the words in all files with names that end with .c
- `ls foo?bar`
  list all filenames that start with foo, followed by an arbitrary character and bar

Hidden Files

Files with names starting with . are hidden, they are not listed nor matched by wildcards

This is why `ls` does not show . nor ..

Useful for avoiding clutter (e.g. many .*rc files in ~)

`ls -a` reveals them

Filename Completion

Many shells have a filename completion feature: when hitting the <tab> key the shell tries to complete the filename. Saves typing!

```
cat super<tab>
```

will complete the command to

```
cat supercalifragilisticexpialidocious
```

if this is the only filename starting with super

Input/Output Redirection

Output of programs can be stored in a file using >:

```
cat file1 file2 > file3
```

[Writes contents of files file1 and file2 to file3]

Generates error message if file3 already exists

Use >! to override

```
cat > foo
```

[copy keyboard input ended by <ctrl-d> to file foo]

Input can also be redirected:

```
grep foo < text
```

[Display all lines in file text that contain foo]

Or both: `sort < file > file.sorted`

Pipes

Powerful UNIX feature: output of commands can become input for subsequent commands

```
grep aaa file | wc -l
```

[count the number of lines in file that contain aaa]

```
sort file | uniq | wc -l
```

[count the number of unique lines in file]

Edit Textfiles

- Many good editors exist: emacs, vi, vim, ...
- emacs is very powerful
- Type `emacs x <ret>` to edit file x
- Large number of commands bound to keys. E.g.
  - <ctrl-x> <ctrl-s> : save buffer
  - <ctrl-x> <ctrl-f> : load file
  - <ctrl-x> <ctrl-c> : exit
  - <ctrl-s> : search
  - <alt-%> : search and replace
  - <ctrl-x> 2 : split window; <ctrl-x> o : switch buffer
  - <alt-x> command : launch external commands such as gdb, gnus
- `man emacs`, emacs reference cards, emacs tutorial (in help menu or on the web)
- Highly customizable: `emacs ~/.emacs`

More Details

Lab 1: UNIX commands

Lab 2: Shell programming and emacs

First C++ Program

- Create file hello.C using emacs and save it

```
// this program prints "hello world" to standard output
#include <iostream>
using namespace std;

int main()
{
  cout << "hello world" << endl;
  return 0;
}
```

- `g++ -o hello hello.C` generates executable `hello` which prints `hello world` after being invoked by issuing `./hello`
- Without the `-o hello` option, g++ creates executable file `a.out`