

CSE2031 Software Tools - UNIX introduction

Summer 2010

Przemyslaw Pawluk
presented by Shakil Khan

Department of Computer Science and Engineering
York University
Toronto

June 29, 2010

Notes

Table of contents

- 1 Introduction to UNIX
- 2 UNIX Shells
- 3 Commands Overview
- 4 grep family

Notes

Our goal

Our goal is to see how we can use Unix as a tool for developing programs

Notes

Processes

- Each running program on a Unix system is called a process
- Processes are identified by a number (process id or PID)
- Usually many processes running simultaneously
- Each process has a unique PID

Notes

Current Working Directory

- Every process has a current working directory
- In a shell, the command `ls` shows the contents of the current working directory
- `pwd` shows the current working directory
- `cd` changes the current working directory

Notes

Path names

- A *path name* is a reference to something in the filesystem
- A path name specifies the set of directories you have to pass through to find a file
- Directory names are separated by `'/'` in Unix
- Path names beginning with `'/'` are absolute path names.
- Path names that do not begin with `'/'` are relative path names (Start search in current working directory)

Notes

Special characters

- `.` means "the current directory"
- `..` means "the parent directory"
- `~` means "home directory"

Try

```
1 ls .  
2 ls ..  
3 ls ~
```

Notes

Devices in UNIX

- `/dev` contains devices
- Look like files but really communicate with devices.

For example:

- `/dev/tty` the terminal (or virtual terminal) you are currently using
- `/dev/zero` an input stream which returns an endless stream of null bytes (`'\0'`)
- `/dev/null` the bitbucket discards any input, generates no output (empty)

Notes

Use of dev/null

To discard stdout of a command:
`cat hello.c >/dev/null`

To provide no input to a command:
`cat </dev/null`

Notes

Shells – intro

What is shell?

Ordinary program which acts as a command interpreter and offers multiple benefits including

- Filename shorthand
- I/O redirection
- Personalizing the environment
- Programming language

Notes

Shells

- There are many programs which are shells
- The most common Unix shells are:
 - Bourne shell (sh)
 - C shell (csh)
 - Korn shell (ksh)
 - Also: Bourneagain shell (bash).
- In this course we are mostly concerned with the Bourne shell (sh)

Notes

How does it work?

When a command is entered shell does

- Process metacharacters
- Command line substitution
- Command execution

Notes

Special Characters

- `>` `>>` `<` `|` – IO redirection
- `*` `?` `[...]` – Filename shorthand
- `'command'` – Command substitution
- `||` `&&` – Conditional execution
- `(...)` – Group commands
- `&` – Background processing
- `<<tok` – Here document
- `$` – Expand value
- `\` `#` `;` – Escape, comment, terminator
- `'` `"` – Single/double quotes

Notes

Commands Overview

- Basic tools: `ls`, `cp`, `mv`, ...
- Advanced tools:
`grep`, `sort`, `cut`, `uniq`, `tr`, `find`, `xargs`, `sed`, `awk`

Notes

Basic Unix Commands(1)

- `ls` list directory content
- `cp` file copy
- `mv` file renaming, moving
- `rm` delete files
- `mkdir` create a new directory
- `cd` change directory
- `pwd` print current working directory
- `cat` print text files
- `more` print text files page by page
- `less` view text files
- `head` print first part of a text file
- `tail` print last part of a text file

Notes

Basic Unix Commands(2)

- `expr` evaluate an expression
- `echo` display a line of text
- `date` print and set system date and time
- `ps` process status
- `kill` kill a process (send a signal)
- `top` display top CPU processes
- `od` octal dump of a file
- `du` disk usage
- `chmod` change file access permission
- `chgrp` change group ownership
- `ln` link files
- `diff` difference of two files
- `basename` base name of a full path name

Notes

Combining Commands

If we just run a command, e.g. `wc` then the terminal is used for `stdin`, `stdout`, and `stderr` by default. However, we don't need to use the terminal. We can control what `stdin`, `stdout`, and `stderr` refer to.

The simple case is redirection using a file

```
wc <foo/bar/file
```

i.e. use the contents of the file "`foo/bar/file`" instead of the terminal for `stdin`

```
wc >foo/bar/file
```

i.e. put the output of `stdout` into "`foo/bar/file`" instead of the terminal

Notes

Pipes

We can also redirect `stdout` or `stdin` to other programs (instead of files)

```
cat myfile | wc
```

this takes the `stdout` of the `cat myfile` command and makes it the `stdin` of the `wc` command

Notes

Shell Scripting

A shell provides

- Basic interactive shell
- Programming environment

Notes

Unix Filters

Filters are a large family of UNIX programs that:

- Read text input line by line
- Perform some transformation
- Write some output

Simple filters: `grep, cut, sort, uniq, tr`
`$ grep 'tom' /etc/passwd`
print lines containing 'tom'

Programmable filters: `awk, sed`
`$ awk '/tom/ {print}' /etc/passwd`

Notes

Design of Filters

- Each processes argument files or stdin if arguments are missing
- Each writes to stdout. Arguments never specify stdout, unless there is an option (e.g. `-o`)
- Some optional arguments (i.e. options, `-a -n ...`) may precede input filename(s)
- Error messages are written to stderr

Notes

grep family

- **grep**: searches text files for pattern and prints all lines that contain that pattern
- **egrep** (expression grep): same as grep but supports full regular expressions
- **fgrep** (fast grep): searches for a string, instead of pattern

Notes

grep

Prints out all lines in the input that match the given regular expression

grep [options] pattern [file ...]

e.g.

grep hello

Prints out all lines of stdin containing "hello"

Exit status

grep exits with a value:

- 0 if pattern found
- 1 if pattern not found
- 2 if file not found

Can be used in scripts!

Notes

Frequently Used Options

- **-i** ignore case of letters (case insensitive search)
- **-v** invert search (print lines that dont match)
- **-c** displays count of matching lines
- **-w** search for expression as distinct word
- **-n** precede each line with line number
- **-l** list only input filenames where matches occur
- **-h** do not display filenames
- **-s** work silently (suppress error messages)

Notes

Regular Expressions

- A regular expression is a special string (like a wildcard pattern)
- A compact way to represent text patterns
- A compact way of matching several text lines with a single string
- Provide a mechanism to select specific strings from a set of character strings

Notes

Basic RE vs. Extended RE

Basic

Letters and numbers are literal that is they match themselves:

e.g. "foobar" matches "foobar"

'.' matches any single character (i.e. exactly one)

[xyz] matches any character in the set (ranges via '-')

[^xyz] matches any character not in the set

'*' matches 0 or more occurrences of last char

'?' matches 0 or 1 occurrences of last char

'+' matches 1 or more occurrences of the last char

'^' matches the beginning of the line

'\$' matches the end of the line

"\<" and "\>" match begin and end of a word

\{n\} matches exactly n occurrences

\{n,\} matches at least n occurrences

\{n,m\} matches occurrences between n and m

Notes

Interesting uses

```
grep -v '^#!'
```

Removes all lines beginning with '#!'

```
grep -v '^[ ]*$'
```

Removes all lines which are either empty or contain only spaces (all empty lines)

```
ls -l | grep "^[^d]"
```

List only files that are not directories

Notes

fgrep

Like `grep`, `fgrep` searches for things but does not do regular expressions just fixed strings.

```
fgrep 'hello.*goodbye'
```

Searches for string "hello.*goodbye" does not match it as a regular expression

Notes

cut

- Used to split lines of a file
- A line is split into fields
- Fields are separated by delimiters
- A common case where a delimiter is a space or tab character
- Default delimiter is tab

Syntax

```
cut [-ffields] [-ccolumns] [-dcharacter] [filename ...]
```

Notes

sort

- sorts lines in a file
- Like before, if no files are given, sorts stdin and writes result to stdout
- By default sorts lines in ascending alphabetical order

Syntax

```
sort [options ... ] [file ... ]
```

Notes

sotr – Options

- `-r` sort in reverse order (descending)
- `-n` treat each line as a number and sort numerically
- `-kN` sort based on the Nth field, e.g. `-k2` or `-k4`
- `-t`: specify field separator (default space and tab)

Notes

uniq

Removes repeated lines in a file

Syntax

`uniq [-c] [input [output]]`

Notice difference in args:

- 1st filename is input file
- 2nd filename is output file

Notes

Notes
