

CSE2031 Software Tools -UNIX introduction

Przemysla Pawluk presented by Shakil

Introduction to UNIX

Commands Overview grep family

1/36

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

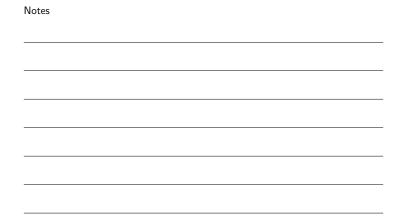




Table of contents

Tools -UNIX introduction Przemysla Pawluk

Introduction to UNIX



2 UNIX Shells



3 Commands Overview

Commands Overview grep family

4 grep family





Our goal

CSE2031 Software Tools -UNIX in-

Przemysla Pawluk presented by Shakil Khan

Introduction to UNIX

Commands Overview Our goal is to see how we can use Unix as a tool for developing programs $% \left\{ 1,2,\ldots ,n\right\}$

Notes			



Processes

CSE2031 Software Tools -UNIX introduction Przemysla Pawluk

Introduction

UNIX Shells

Shells Command Overview

- 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

Tools -UNIX introduction Przemyslav Pawluk presented

Introduction to UNIX

Command Overview

- Every process has a current working directory
- In a shell, the command 1s shows the contents of the current working directory
- pwd shows the current working directory
- cd changes the current working directory

Notes			



Path names

CSE2031 Software Tools -UNIX in-

Przemyslav Pawluk presented by Shakil

Introduction to UNIX

Commands Overview

- $\bullet\,$ 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"

Notes



Devices in UNIX

/dev contains devices

• Look like files but really communicate with devices.

For example:

- \bullet /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

-			

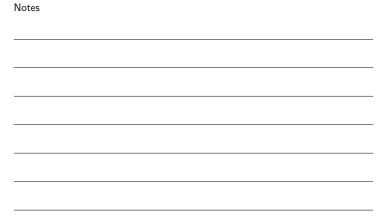


$\mathsf{Shells}-\mathsf{intro}$

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





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

Notes

Notes



How does it work?

When a command is entered shell does

- Process metacharacters
- Command line substitution
- Command execution



Special Characters

• > >> < | - IO redirection

ullet * ? [...] – Filename shorthand

• 'command' - Command substitution

Notes

Notes

• || && - Conditional execution

ullet (...) – Group commands

• & - Background processing

• <<tok – Here document

• \$ - Expand value

• \ # ; - Escape, comment, terminator

 \bullet ' " — Single/double quotes



Commands Overview

• Basic tools: 1s, cp, mv, ...

grep, sort, cut, uniq, tr, find, xargs, sed, awk



Basic Unix Commands(1)

• 1s 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)

• ps process status

• expr evaluate an expression echo display a line of text

• du disk usage • chmod change file access permission • chgrp change group ownership

• kill a process (send a signal) • top display top CPU processes • od octal dump of a file

• date print and set system date and time

• 1n link files

• diff difference of two files

• basename base name of a full path name

Notes



Combining Commands

stderr refer to.

terminal for stdin

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

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

If we just run a command, e.g. wc then the terminal is used for stdin, stdout, and stderr by default. However, we dont need to use the terminal. We can control what stdin, stdout, and

Notes

Notes



Pipes

of fi	les)									
cat	myfil	le		WC						
hic	takes t	tha	ct	dout	of the	cat	myfile	command	and	mal

stdin of the wc command

We can also redirect stdout or stdin to other programs (instead $% \left\{ 1\right\} =\left\{ 1\right\} =\left\{$



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

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)
- \bullet Some optional arguments (I.e. options, -a -n \dots) may precede input filename(s)
- Error messages are written to stderr

-	



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

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

grep [options] pattern [file ...]

Prints out all lines of stdin containing "hello"

• fgrep (fast grep): searches for a string, instead of pattern

Notes		



expression

grep 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
- -1 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
- Provide a mechanism to select specific strings from a set of character strings

ivotes			



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

 $\ref{eq:constraints}$ matches 0 or 1 occurrences of last char

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

'\$' 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

Notes

Notes



- 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

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

• Like before, if no files are given, sorts stdin and writes result

• By default sorts lines in ascending alphabetical order



<u> </u>	/n	ta	<u>×</u>	

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

• sorts lines in a file

to stdout



$\mathsf{sotr}-\mathsf{Options}$

- -r sort in reverse order (descending)
- -n treat each line as a number and sort numerically
- \bullet -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

uniq [-c] [input [output]]
Notice difference in args: $\bullet \ 1st \ filename \ is input \ file$ • 2nd filename is output file

Notes

Notes			