

# UNIX Commands

COMP 444/5201

Revision 1.4

January 25, 2005

# Contents

- Shell Intro
- Command Format
- Shell I/O
- Command I/O
- Command Overview

# Shell Intro

- A system program that allows a user to execute:
  - shell functions (internal commands)
  - other programs (external commands)
  - shell scripts
- Linux/UNIX has a bunch of them, the most common are
  - `tcsh`, an expanded version of `csh` (Bill Joy, Berkley, Sun)
  - `bash`, one of the most popular and rich in functionality shells, an expansion of `sh` (AT&T Bell Labs)
  - `ksh`, Korn Shell
  - `zsh`
  - . . .

# Command Format

- Format: command name and 0 or more arguments:

`% commandname [arg1] ... [argN]`

- By % sign I mean prompt here and hereafter.
- Arguments can be
  - options (switches to the command to indicate a mode of operation) ; usually prefixed with a hyphen (-) or two (--) in GNU style
  - non-options, or operands, basically the data to work with (actual data, or a file name)

# Shell I/O

- Shell is a “power-user” interface, so the user interacts with the shell by typing in the commands.
- The shell interprets the commands, that may produce some results, they go back to the user and the control is given back to the user when a command completes (in general).
- In the case of external commands, shell executes actual programs that may call functions of the OS kernel.
- These system commands are often wrapped around a so-called system calls, to ask the kernel to perform an operation (usually privileged) on your behalf.

# Command I/O

- Input to shell:
  - Command name and arguments typed by the user
- Input to a command:
  - Keyboard, file, or other commands
- Standard input: keyboard.
- Standard output: screen.
- These `STDIN` and `STDOUT` are often together referred to as a terminal.
- Both standard input and standard output can be redirected from/to a file or other command.
- File redirection:
  - `<` input
  - `>` output
  - `>>` output append

# Commands

July 10, 2003

Serguei A. Mokhov,  
mokhov@cs.concordia.ca

7

# man

- Manual Pages
- The first command to remember
- Contains info about almost everything :-)
  - other commands
  - system calls
  - c/library functions
  - other utils, applications, configuration files
- To read about man itself type:  
% man man
- **NOTE:** unfortunately there's **no**  
% man woman ...

# which

- Displays a path name of a command.
- Searches a path environmental variable for the command and displays the absolute path.
- To find which `tcsh` and `bash` are actually in use, type:
  - `% which tcsh`
  - `% which bash`
- `% man which` for more details

# chsh

- Change Login Shell
- Login shell is the shell that interprets commands after you logged in by default.
- You can change it with chsh (provided that your system admin allowed you to do so).
- To list all possible shells, depending on implementation:  
% chsh -l  
% cat /etc/shells
- % chsh with no arguments will prompt you for the shell.

# whereis

- Display all locations of a command (or some other binary, man page, or a source file).
- Searches all directories to find commands that match `whereis`' argument
- `% whereis tcsh`

# General Commands

July 10, 2003

Serguei A. Mokhov,  
mokhov@cs.concordia.ca

12

# passwd

- Change your login password.
- A very good idea after you got a new one.
- It's usually a paranoid program asking your password to have at least 6 chars in the password, at least two alphabetical and one numerical characters. Some other restrictions (e.g. dictionary words or previous password similarity) may apply.
- Depending on a privilege, one can change user's and group passwords as well as real name, login shell, etc.
- `% man passwd`

# date

- Guess what :-)
- Displays dates in various formats
- `% date`
- `% date -u`
  - in GMT
- `% man date`

# cal

- Calendar
    - for month
    - entire year
  - Years range: 1 - 9999
  - No year 0
  - Calendar was corrected in 1752 - removed 11 days
- |                |                     |
|----------------|---------------------|
| • % cal        | current month       |
| • % cal 2 2000 | Feb 2000, leap year |
| • % cal 2 2100 | not a leap year     |
| • % cal 2 2400 | leap year           |
| • % cal 9 1752 | 11 days skipped     |
| • % cal 0      | error               |
| • % cal 2002   | whole year          |

# clear

- Clears the screen
- There's an alias for it: **Ctrl+L**
- Example sequence:
  - % cal
  - % clear
  - % cal
  - Ctrl+L

# sleep

- “Sleeping” is doing nothing for some time.
- Usually used for delays in shell scripts.
- `% sleep 2` 2 seconds pause

# Command Grouping

- Semicolon: “;”
- Often grouping acts as if it were a single command, so an output of different commands can be redirected to a file:
- `% (date; cal; date) > out.txt`

# alias

- Defined a new name for a command
- `% alias`
  - with no arguments lists currently active aliases
- `% alias newcommand oldcommand`
  - defines a newcommand
- `% alias cl cal 2003`
- `% cl`

# unalias

- Removes alias
- Requires an argument.
- `% unalias cl`

# history

- Display a history of recently used commands
- `% history`
  - all commands in the history
- `% history 10`
  - last 10
- `% history -r 10`
  - reverse order
- `% !!`
  - repeat last command
- `% !n`
  - repeat command **n** in the history
- `% !-1`
  - repeat last command = !!
- `% !-2`
  - repeat second last command
- `% !ca`
  - repeat last command that begins with 'ca'

# apropos

- Search man pages for a substring.
- % `apropos word`
- Equivalent:
- % `man -k word`
- % `apropos date`
- % `man -k date`
- % `apropos password`

# exit / logout

- Exit from your login session.
- % `exit`
- % `logout`

# shutdown

- Causes system to shutdown or reboot cleanly.
- May require superuser privileges
- % `shutdown -h now` - stop
- % `shutdown -r now` - reboot

# Files

July 10, 2003

Serguei A. Mokhov,  
mokhov@cs.concordia.ca

25

# ls

- List directory contents
- Has whole bunch of options, see `man ls` for details.
- `% ls`
  - all files except those starting with a “.”
- `% ls -a`
  - all
- `% ls -A`
  - all without “.” and “..”
- `% ls -F`
  - append “/” to dirs and “\*” to executables
- `% ls -l`
  - long format
- `% ls -al`
- `% ls -lt`
  - sort by modification time (latest - earliest)
- `% ls -ltr`
  - reverse

# cat

- Display and concatenate files.
- `% cat`
  - Will read from STDIN and print to STDOUT every line you enter.
- `% cat file1 [file2] ...`
  - Will concatenate all files in one and print them to STDOUT
- `% cat > filename`
  - Will take whatever you type from STDIN and will put it into the file `filename`
- To exit `cat` or `cat > filename` type `Ctrl+D` to indicate EOF (End of File).

# more / less

- Pagers to display contents of large files page by page or scroll line by line up and down.
- Have a lot of viewing options and search capability.
- Interactive. To exit: 'q'

# less

- `less` ("less is more") a bit more smart than the `more` command
- to display contents of a file:
  - `% less filename`
- To display line numbers:
  - `% less -N filename`
- To display a prompt:
  - `% less -P"Press 'q' to quit" filename`
- Combine the two:
  - `% less -NP"Blah-blah-blah" filename`
- For more information:
  - `% man less`

# touch

- By *touching* a file you either create it if it did not exist (with 0 length).
- Or you update its last modification and access times.
- There are options to override the default behavior.
- `% touch file`
- `% man touch`

# cp

- Copies files / directories.
- % `cp [options] <source> <destination>`
- % `cp file1 file2`
- % `cp file1 [file2] ... /directory`
- Useful option: `-i` to prevent overwriting existing files and prompt the user to confirm.

# mv

- Moves or renames files/directories.
- `% mv <source> <destination>`
  - The `<source>` gets removed
- `% mv file1 dir/`
- `% mv file1 file2`
  - rename
- `% mv file1 file2 dir/`
- `% mv dir1 dir2`

# rm

- Removes file(s) and/or directories.
- `% rm file1 [file2] ...`
- `% rm -r dir1 [dir2] ...`
- `% rm -r file1 dir1 dir2 file4 ...`

# script

- Writes a log (a typescript) of whatever happened in the terminal to a file.
- `% script [file]`
- `% script`
  - all log is saved into a file named typescript
- `% script file`
  - all log is saved into a file named file
- To exit logging, type:
  - `% exit`

# find

- Looks up a file in a directory tree.
- `% find . -name name`
- `% find . \(-name 'w*' -or -name 'W*' \)`

# mkdir

- Creates a directory.
- `% mkdir newdir`
- Often people make an alias of `md` for it.

# cd

- Changes your current directory to a new one.
- `% cd /some/other/dir`
  - Absolute path
- `% cd subdir`
  - Assuming `subdir` is in the current directory.
- `% cd`
  - Returns you to your home directory.

# pwd

- Displays personal working directory, i.e. your current directory.
- % pwd

# rmmdir

- Removes a directory.
- `% rmmdir dirname`
- Equivalent:
  - `% rm -r dirname`

# ln

- Symbolic link or a “shortcut” in MS terminology.
- `% ln -s <real-name> <fake-name>`

# chmod

- Changes file permissions
- Possible invocations
  - `% chmod 600 filename`
  - `-rw----- 1 user group 2785 Feb 8 14:18 filename`  
(a bit not intuitive where 600 comes from)
  - `% chmod u+rw filename`  
(the same thing, more readable)
  - For the assignment:
    - `% chmod u+x myshellscript`  
(myshellscript is now executable)
    - `-rwx----- 1 user group 2785 Feb 8 14:18 myshellscript`

# grep

- Searches its input for a pattern.
- The pattern can be a simple substring or a complex regular expression.
- If a line matches, it's directed to `STDOUT`; otherwise, it's discarded.
- `% echo "blah-foo" | grep blah`
  - Will print the matching line
- `% echo "blah-foo" | grep zee`
  - Will not.
- See a separate `grep` tutorial.

# Pipes

- What's a pipe?
  - is a method of interprocess communication (IPC)
  - in shells a '|' symbol used
  - it means that the output of one program (on one side of a pipe) serves as an input for the program on another end.
  - a set of "piped" commands is often called a pipeline
- Why it's useful?
  - Because by combining simple OS utilities one can easily solve more complex tasks

# More on UNIX Commands and Editors

- <http://www.cs.concordia.ca/help/>