CSE 314: OS Sessional

Shell Commands

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology
Dhaka-1000, Bangladesh

`man`

- format and display the on-line manual pages
- `man name`

`info`

- read Info documents
- `info name`

`su`

- run a shell with substitute user
- `su [OPTION]... [-] [USER [ARG]...]`
- Change the effective user id and group id to that of USER.
- `-`
  make the shell a login shell

`passwd`

- Update A user’s authentication tokens(s)
- `passwd [username]`

echo

- Display a line of text
- `echo [OPTION]... [STRING]...`

`ls`

- List information about the files
- `ls [OPTION]... [FILE]...`
- `-a, --all`
  do not hide entries starting with .
- `-A, --almost-all`
  do not list implied `.` and `..`
- `-h, --human-readable`
  print sizes in human readable format
- `-l`
  use a long listing format
- `-S`
  sort by file size

pwd

- print name of current/working directory
- pwd

cd

- Change the current directory to dir.
- `cd [dir]`
- The variable HOME is the default dir.

`mkdir`

- Make directories
- `mkdir [OPTION] DIRECTORY`
- Create the `DIRECTORY`(ies), if they do not already exist
- `-p, --parents`
  no error if existing, make parent directories as needed
- `-v, --verbose`
  print a message for each created directory

cp

- Copy files and directories.
- `cp [OPTION]... SOURCE DEST`
- `cp [OPTION]... SOURCE... DIRECTORY`
- Copy `SOURCE` to `DEST`, or multiple `SOURCE`(s) to `DIRECTORY`
- `-i, --interactive`
  prompt before overwrite
- `--parents`
  append source path to `DIRECTORY`
- `-R, -r, --recursive`
  copy directories recursively

`cp` — continued

- `-u, --update`
  copy only when the `SOURCE` file is newer than the destination file or when the destination file is missing
- `-v, --verbose`
  explain what is being done

`rm`

- Remove files or directories
- `rm [OPTION]... FILE...`
- `-f`, `--force`
  ignore nonexistent files, never prompt
- `-i`, `--interactive`
  prompt before any removal
- `-r`, `-R`, `--recursive`
  remove the contents of directories recursively
- `-v`, `--verbose`
  explain what is being done

mv

- Move (rename) files
- `mv [OPTION]... SOURCE DEST`
- `mv [OPTION]... SOURCE... DIRECTORY`
- Rename `SOURCE` to `DEST`, or move `SOURCE`(s) to `DIRECTORY`
- `-i, --interactive`
  prompt before overwrite
- `-u, --update`
  move only when the `SOURCE` file is newer than the destination file or when the destination file is missing
- `-v, --verbose`
  explain what is being done

`rename`

- Rename files
- `rename from to file`
- rename will rename the specified files by replacing the first occurrence of `from` in their name by `to`

`ln`

- Make links between files
- `ln [OPTION]... TARGET [LINK_NAME]`
- `ln [OPTION]...  TARGET...  DIRECTORY`
- Create a link to the specified `TARGET` with optional `LINK_NAME`.
- If `LINK_NAME` is omitted, a link with the same basename as the `TARGET` is created in the current directory.
- When using the second form with more than one `TARGET`, the last argument must be a directory; create links in `DIRECTORY` to each `TARGET`.
- `-s, --symbolic`
  make symbolic links instead of hard links

Soft and Hard Links

- Both of these provide a certain measure of dual reference – if you edit the contents of the file using any name, your changes will affect both the original name and either a hard or soft new name.
- The differences between them occurs when you work at a higher level.
- The advantage of a hard link is that the new name is totally independent of the old name – if you remove or rename the old name, that does not affect the hard link, which continues to point to the data while it would leave a soft link hanging pointing to the old name which is no longer there.
- The advantage of a soft link is that it can refer to a different file system (since it is just a reference to a file name, not to actual data.)

pushd

- Moves to a directory pushing the current one to stack
- `pushd [DIR]`
- Adds a directory to the top of the directory stack, or rotates the stack, making the new top of the stack the current working directory.

popd

- Moves to the directory at the top of the stack as well as removes the topmost entry
- popd
- Removes entries from the directory stack. Removes the top directory from the stack, and performs a cd to the new top directory.

dirs

- `dirs`
- Displays the list of currently remembered directories.
- The default display is on a single line with directory names separated by spaces.
- Directories are added to the list with the `pushd` command, the `popd` command removes entries from the list.

`file`

- Determine file type
- `file [-z] file`
- File tests each argument in an attempt to classify it. This causes the file type to be printed.
- `-z`
  Try to look inside compressed files.

`cat`

- Concatenate files and print on the standard output
- `cat [OPTION] [FILE]...`
- `-n, --number`
  number all output lines
- `-s, --squeeze-blank`
  never more than one single blank line

`more`

- File perusal filter for CRT viewing
- `more [file ...]`
- More is a filter for paging through text one screenful at a time.
- Interactive command h `or ?`
  Help: display a summary of these commands. If you forget all the other commands, remember this one.
- Interactive command SPACE
  Display next k lines of text. Defaults to current screen size.

`more` — continued

- Interactive command `RETURN`
  Display next k lines of text. Defaults to `1`. Argument becomes new default.
- Interactive command `q` or `Q` or `INTERRUPT`
  Exit.

`less`

- Opposite of more
- `less [+N] [filename]`
- Less is a program similar to more, but which allows backward movement in the file as well as forward movement.
- Also, less does not have to read the entire input file before starting, so with large input files it starts up faster
- `-N` or `--LINE-NUMBERS`
  Causes a line number to be displayed at the beginning of each line in the display.

`less` — continued

- Interactive command `h` or `H`
  Help: display a summary of these commands. If you forget all the other commands, remember this one
- Interactive command `SPACE` or `^V` or `f` or `^F`
  Scroll forward one window
- Interactive command `b` or `^B` or `ESC-v`
  Scroll backward one window
- Interactive command `g` or `<` or `ESC-<`
  Go to the beginning of file)
- Interactive command `G` or `>` or `ESC->`
  Go to the end of the file.

`less` — continued

- Interactive command `/pattern`
  Search forward in the file for the line containing the pattern.
  - n Go to the next occurrence of pattern
- Interactive command `:n`
  Examine the next file from the list of files given in the command line.
- Interactive command `:p`
  Examine the previous file in the command line list.
- Interactive command `:x`
  Examine the first file in the command line list.
- Interactive command q or `Q` or q or `:Q` or ZZ
  Exits less.

`tail`

- Output the last part of files
- `tail [OPTION]... [FILE]...`
- Print the last 10 lines of each FILE to standard output
- With more than one `FILE`, precede each with a header giving the file name
- With no `FILE`, or when `FILE` is `-`, read standard input
- `-n`, `--lines=N`
  output the last `N` lines, instead of the last 10
- `-f`
  output appended data as the file grows
- `--retry`
  keep trying to open a file even if it is inaccessible when tail starts or if it becomes inaccessible later – useful only with `-f`

`sort`

- sort lines of text files
- `sort [OPTION]... [FILE]...`
- Write sorted concatenation of all `FILE`(s) to standard output.
- `-d`, `--dictionary-order`
  consider only blanks and alphanumeric characters
- `-f`, `--ignore-case`
  fold lower case to upper case characters

wc

- Print the number of bytes, words, and lines in files
- `wc [OPTION]... [FILE]...`
- Print byte, word, and newline counts for each `FILE`, and a total line if more than one `FILE` is specified. With no `FILE`, or when `FILE` is `-`, read standard input.
- `-l`, `--lines`
  print the newline counts
- `-w`, `--words`
  print the word counts

Standard input, output and error

- Most shell commands (and most useful programs) get some type of input, then process the input, and produce some output including error messages.
- Input, output and error messages can have different sources.
- For example:
  - input - may be an internal variable, keyboard, network card, file (and others).
  - output - send to screen, printer, speakers, file (and others).
  - error messages - often to the screen, or a file.

Standard input, output and error — continued

- By default, three default files known as standard files are automatically opened when a command is executed.
- The standard files are standard input (stdin), standard output (stdout), and standard error (stderr).
- By default, the keyboard is standard input and the terminal window is standard output and standard error.
- For example, the command `ls -a` scans the current directory and collects a list of all the files, produces a human readable list, and outputs the result to the terminal window.

Redirection, Piping

- Linux redirection features can be used to detach the default files from stdin, stdout, and stderr and attach other files to them.
- Input redirection:
  - < (less-than symbol) - get input from file instead of the keyboard
- Output redirection:
  - > (greater-than symbol) - send output to file instead of the terminal window
- Append output:
  - >> - command is used to append to a file if it already exists

Redirection, Piping — continued

- The input of a command may come from the output of another command.
- This is accomplished with the `|` pipe operator.

bg

- Resume the suspended job jobspec in the background, as if it had been started with &.

Using Multiple Commands

- Linux allows you to enter multiple commands at one time.
- You separate the commands with a semicolon

`for` Loop

```
for { variable name } in { list }
  do
     execute one for each item in the
     list until the list is
     not finished (and repeat all
     statement between do and done)
  done
```

Editors

- My order of preference
  - xemacs/emacs
  - gedit
  - nedit