Operating Systems

OS C372 & IS C362

Lab Test

S.K. Sahay

BITS-Pilani, Goa Campus, Goa

Glob expressions

- ls [A-M]*; restricting file listings to within the ranges of certain characters.

- [a-dJW-Y]*; matches all files beginning with a, b, c, d, J, W, X or Y

- *[a-d]id; matches all files ending with aid, bid, cid or did

- *.cpp, .c, .cxx; matches all files ending in .cpp, .c or .cxx.

The above way of specifying a file name is called a glob expression.

Linux utility: find

Searching for files:

- find -type d: Shows only directories and not the files they contain.
- find -type f: Shows only files and not the directories that contain them, even though it will still descend into all directories.
- find -name <filename>: Finds only files that have the name <filename>. e.g. find -name '*.c'
- find -size [+/-]<size>: Finds only files that have a size larger/smaller (+/-) or than <size> kilobytes.

Linux utility: find

Searching for files:

- find <directory> [<directory> ...]: Starts find in each of the specified directories.

- With -exec option causes find to execute a command for each file it finds, for e.g:
  - find /usr -type f -exec ls '-al' '{}' ';'

Linux utility: grep

Print lines matching a pattern

- grep [options] <pattern> <filename> [<filename> ...]
- grep -n <pattern> <filename>: Shows the line number in the file where the word was found.
- grep -<num> <pattern> <filename>: Prints out <num> of the lines that came before and after each of the lines in which the word was found.
- grep -A <num> <pattern> <filename>: Prints out <num> of the lines that came after each of the lines in which the word was found.

Linux utility: grep

Searching within files

- grep -B <num> <pattern> <filename>: Prints out <num> of the lines that came before each of the lines in which the word was found.

- grep -v <pattern> <filename>: Prints out only those lines that do not contain the word you are searching for.

- grep -i <pattern> <filename>: Does the same as an ordinary grep but is case insensitive.

Regular Expressions

- A regular expression is a sequence of characters that forms a template used to search for strings & words, phrases, or just about any sequence of characters.

- It is a search pattern used for line-by-line searches.

- With regular expressions, the wildcard (?) to use is the . character.

Regular Expressions

Examples:

- grep -w ’t[a-i]e’ Matches the words tee, the, and tie. Bracket mean to match one character that can be anything between a to i.
- grep -w ’cr[a-m]*t’ Matches the words craft, credit, and cricket. The * means to match any number of the previous character.
- grep -w ’kr.*n’ Matches the words kremlin and krypton. The . matches any character and the * means to match the dot any number of times.

Regular Expressions

Examples:

- egrep -w ’(th|sh).*rt’ Matches the words shirt, short, and thwart.
- grep -w ’thr[aeiou]*t’ Matches the words threat and throat.
- grep -w ’thr[^a-f]*t’ Matches the words throughput and thrust.
- characters . \ [ ] * + ? cannot be used to match characters. However use of \., forces interpretation as an actual i.e. regular expression myfile\.txt will match only myfile.txt.

Regular Expressions

Examples:

- grep -w ’th[a-t]\2,3\t’ Matches the words theft, thirst, threat, thrift, and throat.

- grep -w ’th[a-t]\4,5\t’ Matches the words theorist, thicket, and thinnest.

- grep ’trot’ Matches the words electrotherapist, betroth, and so on, but

- grep ’\ <trot\ >’ matches only the worrd trot.

- grep -w ’(this—that—c[aeiou]*t)’ Matches the words this, that, cot, coat, cat, and cut.

Pipe and Redirection operator

- Pipe ‘|’ is is one of the powers of LINUX /UNIX.
- Places the data in the one end of a funnel while another program reads that data from the other end.
- It allow two separate programs to perform simple communications with each other.
- grep GNU myfile.txt | grep Linux
  Often used multiple times for filtering.
- A complex piping

  ```
  cat /bin/cp | strings | tr 'A-Z' 'a-z' | grep '^[a-z]' | sort -u > Ascii.txt
  ```

Linux utility: cc, ar & ranlib

- Creating object file.

```
cc -Wall -c simple_sqrt.c
cc -Wall -c simple_pow.c
```

- To archive the object files into a library.

```
ar libmath.a simple_sqrt.o simple_pow.o
```

- For indexing the archive.

```
ranlib libmath.a
```

Linux utility: make

- A utility that only recompiles object files whose sources have changed.
- Makefiles contain lists of rules and dependencies describing how to build a program.
- Inside a Makefile you need to state a list of what-depends-on-what dependencies, as well as the shell commands needed to achieve each goal.

```
mytest:  libmath.a mytest.o
cc -Wall -o $@ mytest.o -L. -lmath
```

Linux utility: make

- The full Makefile rule is

```
libmath.a:  simple_sqrt.o simple_pow.o
     rm -f $@
     ar rc $@ simple_sqrt.o simple_pow.o
     ranlib $@
```

- Short way of stating such a rule

```
.c.o:
          cc -Wall -c -o $*.o $<
```

Linux utility: make

- Putting it all together

```
# Comments start with a # (hash).
# Makefile to build limmath.a and mytest program.

OBJS = simple_sqrt.o simple_pow.o
LIBNAME = math
CFLAGS = -Wall

all:  lib$(LIBNAME).a mytest

mytest:  lib$(LIBNAME).a mytest.o
       cc $(CFLAGS) -o $@ mytest.o -L. \
       -l$LIBNAME
```

Linux utility: make

```
lib$(LIBNAME).a:   $(OBJS)
        rm -f $@
        ar rc $@ $(OBJS)
        ranlib $@

.c.o:
        cc $(CFLAGS) -c -o $*.o $<

clean:
        rm -f *.o *.a mytest
```

Linux utility: make

- Type `make` in the current directory to built everything.
- Type `make clean` to remove all built files.
- Makefiles have far more uses than just building C programs.
- Anything needs to be built from sources can employ Makefiles to make things easier.