Week 1

------

commands (whoami, whereis, find, etc)

man pages

unix file permissions

piping and redirection

find . -name "\*o" finds all files ending in "o". recursively checks all directories

find . -maxdepth 1 -name "\*o" limits directory depth to one (only current)

find . -type f/d/l find files by type

find . -type f -atime +30 access time. any files that have not been accessed for 30 days or more

find . -type f -perm 664 finds files with permissions 664

9 bit permission

- --- --- ---

type(file, directory) user (rwx) group other

bits are 0 or 1 so -110110100 = 664

chmod u+x example.txt (adds execute permission for user on file example.txt)

u-x removes execute permission

u=x set perm to...

chmod 700 example.txt (-111000000) user gets all, everyone else has nothing

man pages:

1. name and short description

2. synopsis, how command is used, syntax cp[options]....source dest

3. description heading: options/flags, how to use, what they do -r: recursive

> symbol only redirects stdout

ly > out.txt 2 >&1 writes "ly:command not found" to out.txt. redirects stderr

echo "hello world" >> out.txt append operator

cat page.html | tr -c 'A-Za-z' '[ln\*]' | sat -u | comm -23 -words

this dash refers to stdin

week 2

------

regular expressions and wildcards

grep, sed, tr

bash scripting

single, double, back ticks

echo sunday | sed 's/day/night/'

regex replacement string pattern is also OK to put here

output is sunnight. s=replace. replaces day with night

cat script | sed 's/^#.\*//' remove comments from a bash script

echo abba | sed 's/a\(..\)a/\1/' output is bb \1 refers to group 1, namely \(..\)

match find a..a and replace with ..

fgrep 'fixed string'

grep basic regular expression special chars must be backslashed

egrep reg exp understands special chars

cat program.c | egrep '^//(todo|fixme)'

cat program.c | grep '^//\(todo\|fixme\)'

bash script:

first line: #! /bin/bash

rest: commands

var =8 $var

if[5 -gt 1]

then....

else....

fi

for word in $file

do........

done

IFS=\n changes default for loop seperator to \n (regular default is ' ')

while[$count -gt 0]

do

.......

done

$? this is the return value of previously run command

$# this is number of args

$1 this is arg number one

3/13

week 3

------

compilation process

build process

python

patch, diff

source code (cpp, c, h)------>pre-processor---->extended source code---->complex---->assembly file (s)----->assembler---->object file (o)---->linker---->executable

Makefile

has a Target with prereqs/dependency list

then the command/rule

e.g.

file.o: file.c file.h

gcc -g -Wall file.c

this command compares target to prereqs, it will compare modification times and run command if target is older than prereqs

clean:

rm -rf \*.o a.out

check:

probably some compare command

build process

configure

-generates Makefile

-checks dependencies

make

-compiles code (see above)

make install

-copies executable to system directories

patching

how to make a patch: diff -u orig new > patch.txt

how to patch: patch -pNum < patch.txt

python

starts with #!/bin/python

import sys

lists are dynamic and heterogeneous

list=[150, "hello", 2.43]

for x in list:

print i

output is 150 hello 2.43

for i in range(len(list)):

print i

output is 0 1 2

python doesnt use brackets or keywords. USE INDENTATION

how to check # of args

if len(sys.argv)!=2

usage(sys.argv[0]+'accepts 1 arguments')

does count name of script as arg

script arg1 (2)

in bash

if [ $# -ne 2 ]

does not count name of script as arg

script arg1 arg2 (2)

functions and classes

def usage(msg):

sys.stderr.write('Error'+msg+"\n")

sys.exit(1)

class complex

def \_\_init\_\_(self, real, imag):

self.r=real

self.i=imag

x=complex(3.0, -4.5)

print x.r //3.0

print x.i //-4.5

week 4

------

version control systems (git)

git commands

4 object types:

blob, tree, commit, tag

commands:

checkout, add, commit

init, clone

week 5

------

c programming

gdb and debugging

know function pointers and file i/o and memory allocation

gcc -g file.c

gdb executable

run args

set breakpoints

break line#

continue, next, step

week 6

------

ssh

public/private keys

digital signatures

document--hash func-->hash---private key--->encrypted hash (signature)

get doc with signature, seperate and decrypt using public key and hash func