Warning: These notes are not complete, it is a Skelton that will be modified/add-to in the class. If you want to us them for studying, either attend the class or get the completed notes from someone who did

CSE2301

Unix/Linux Introduction

These slides are based on slides by Prof. Wolfgang Stuerzlinger at York University

Introduction

- In this part, we introduce
  - OS (Linux)
  - File system
  - Shell commands
  - Pattern matching
  - Shell programming

Unix

- What does an OS do?
  - File management
  - Scheduling
  - Memory management
  - I/O management
- Examples

Unix

- OS includes
- Kernel: Performs key OS functions
- System programs: various tools
- Shell: Interface to the user

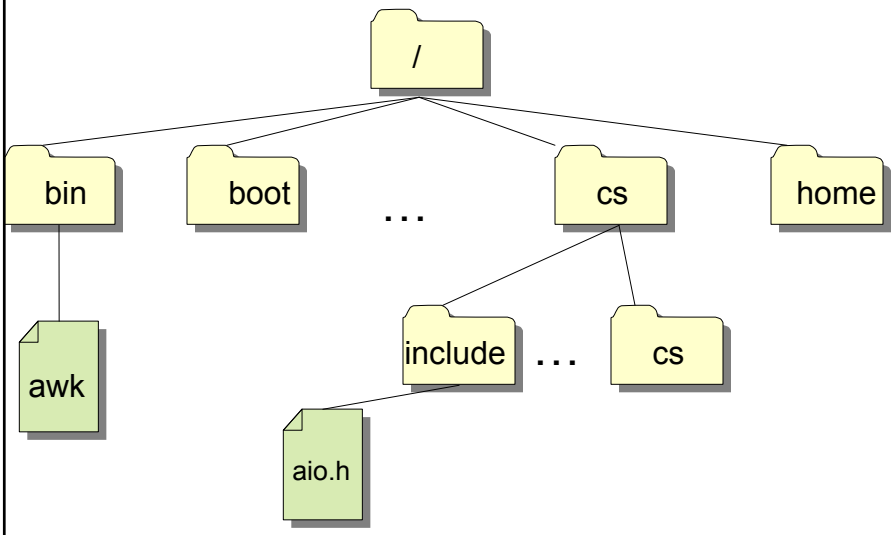
Processes

- Each program running is called a process
- Each process has its own identification PID
- If the program is running twice, even by the same user, these are 2 different processes.

File System

- In Unix, the files are organized into a tree structure with a root named by the character '/'.
- Everything in the file system is a file or subdirectory

Our File System



File System

- File names could be relative (with respect to the current directory) or using full path name (relative to /) for example aio.h or /cs/include/aio.h
- Your home directory is ~username, so in my case ~aboelaze/test.c is equivalent to /cs/home/aboelaze/test.c

Devices

- /dev contains devices, just like any other file (fopen, fread, fwrite, ..) but it communicate with a device.
- /dev/tty
- /dev/null
- /dev/zero

Unix Commands

- ls cp mv rm mkdir cd pwd cat less more head tail ….
- bg, fg, CTRL-C, CTRL-Z
- kill ps od diff ln echo …
- Redirection and pipes Examples

- tigger 215 % ls –las
- total 44
- 4 drwx------ 2 aboelaze faculty 4096 Nov 29 13:44 ./
- 4 drwx------ 9 aboelaze faculty 4096 Nov 29 14:47 ../
- 4 -rw------- 1 aboelaze faculty 184 Nov 18 13:30 data
- 4 -rw------- 1 aboelaze faculty 23 Nov 28 19:52 file1
- 4 -rw------- 1 aboelaze faculty 24 Nov 28 19:52 file2
- 4 -rw------- 1 aboelaze faculty 481 Nov 29 12:27 mergefiles.awk
- 4 -rw------- 1 aboelaze faculty 178 Nov 28 19:32 p1
- 4 -rw------- 1 aboelaze faculty 1245 Nov 18 13:29 prchecks.awk
- 4 -rw------- 1 aboelaze faculty 83 Nov 14 17:46 t
- 4 -rwx------ 1 aboelaze faculty 35 Nov 21 13:08 test.sh*
- 4 -rw------- 1 aboelaze faculty 50 Nov 1 18:31 unmatched
- chmod 744 file What does it mean?
- chmod [ugo][+-][rwx] chmod ug+rw p1

Shell Pattern Matching--Wild Cards

- The character * matches any string of characters
- ? Matches a single character
- [0-9] matches any digit
- [a-z] matches any small case letter
- \c matches c only
- a|b matches a or b **in case expression only**

Shell Variables

- set x = 3 -- csh
- x=3 -- sh
- echo x
- echo $x what is the difference

PATH path

- The shell searches in PATH looking for the command you typed
- echo $PATH .:/usr/local/bin:/usr/ucb: /usr/bin /usr/etc:/etc:/bin:/usr/bin/X11
- set path = ($path /a/b/c) --csh
- PATH=$PATH:/a/b/c --sh
- Aliases and startup files

Shell scripting

```
#!/cs/local/bin/sh
echo “Hello World”
```

```
tigger 397 % script1
Hello World
tigger 398 %
```

```
echo  -n “Hello
World”
```

```
tigger 393 % script1
Hello Worldtigger 394 %
```

```
#!/cs/local/bin/sh
echo "Now I will guess your OS"
echo  -n "Your OS is : "
uname
```

```
tigger 399 % script1
Now I will guess your OS
Your OS is : Linux
tigger 400 %
```

Shell Scripting

```
#!/cs/local/bin/sh
echo -n "Please enter your first name : "
read FNAME
echo -n "Last name pelase : "
read LNAME
MESSAGE=" Your name is : $LNAME , $FNAME"
echo "$MESSAGE"
```

```
tigger 439 % script3
Please enter your first name : Mokhtar
Last name pelase : Aboelaze
 Your name is : Aboelaze , Mokhtar
```

Shell Scripting

```
#!/cs/local/bin/sh
read FNAME
echo "1-> $FNAME123"
echo "2-> ${FNAME}123"
```

```
tigger 454 % script4
abcd
1->
2-> abcd123
tigger 455 %
```

Shell Scripting

```
# Set the initial value.
myvar=abc
echo "Test 1 ======"
echo $myvar       # abc
echo ${myvar}     # same as above, abc
echo {$myvar}     # {abc}
echo "Test 2 ======"
echo myvar        # Just the text myvar
echo "myvar"      # Just the text myvar
echo "$myvar"     # abc
echo "\$myvar"    # $myvar
echo "Test 3 ======"
echo $myvardef    # Empty line
echo ${myvar}def  # abcdef
```

```
$ sh var_refs
Test 1 ======
abc
abc
{abc}
Test 2 ======
myvar
myvar
abc
$myvar
Test 3 ======

abcdef
```

Shell Scripting

```
echo "Test 4 ======"
echo $myvar$myvar      # abcabc
echo ${myvar}${myvar}  # abcabc
echo "Test 5 ======"
# Reset variable value, with spaces
myvar=" a   b    c"
echo "$myvar"         # a    b    c
echo $myvar           # a b c
```

```
Test 4 ======
abcabc
abcabc
Test 5 ======
a    b    c
a b c
```

Looping

- for *variable* in *list_of_items*
- do
- *command1*
- *command2*
- ...
- *last_command*
- done

Looping

- for filename in *
- do
- echo $filename
- done

- for filename in *.doc
- do
- echo "Copying $filename to $filename.bak"
- cp $filename $filename.bak
- done

Looping

- for i in 1 2 3 4 5 6 7 8 9 10
- do
- echo -n "...$i"
- done
- echo # Clean up for next shell prompt

Looping

```
# Counts by looping for a fixed number of times
# Note do on same line requires semicolon.
for i in 1 2 3 4 5 6 7 8 9 10; do
   echo -n "...$i"
done
echo    # Output newline
```

```
Counts by looping for a fixed number of times
# Note do on same line requires semicolon.
for i in 1 2 3 4 5 6 7 8 9 10; do
   echo -n "...$i"
done
sleep 5
echo    # Output newline
```

Looping

```
# Counts backwards
for i in 10 9 8 7 6 5 4 3 2 1
do
    echo -n "...$i"
done
echo    # Output new line
echo "Blast off!"
$ sh counter2
...10...9...8...7...6...5...4...3...2...1
Blast off!
```

Looping

```
# C-language-like for loop.
# Must be run with bash.
max=10
for ((i=1; i <= max ; i++))
do
    echo -n "$i..."
done
echo
```

If-then-else

```
if (condition_command) then
    command1
    command2
    ...
    last_command
else
    command1
    command2
    ...
    last_command
fi
```

- `exit 0` ← return0
- `exit 1` ← return1
- `if (sh return0) then`
- `   echo "Command returned true."`
- `else`
- `    echo "Command returned false."`
- `fi`

Redirection

- What is that?
- ls /fred > /dev/null 2> /dev/null

If elif

```
echo –n “checking for a C shell: “
if(which csh >/dev/null 2> /dev/null) then
     echo “csh found.”
elif (which csh >/dev/null 2> /dev/null) then
     echo “tcsh found, which works with csh”
else
     echo “csh not found”
fi
```

Example

- Mycal program In class discussion

```
#!/cs/local/bin/sh

case $# in
0)    set `date`; m=$2; y=$6;;
1)    m=$1; set `date`; y=$6;;
2)    m=$1; y=$2;;
esac

case $m in
jan*|Jan*)      m=1;;
feb*|Feb*)      m=2;;
mar*|Mar*)      m=3;;
apr*|Apr*)      m=4;;
may*|May*)      m=5;;
jun*|Jun*)      m=6;;
jul*|Jul*) m=7;;
aug*|Aug*)      m=8;;
sep*|Sep*)      m=9;;
oct*|Oct*)      m=10;;
nov*|Nov*)      m=11;;
dec*|Dec*)      m=12;;
[1-9]|10|11|12) ;;
*)          y=$m;m="";;
esac
/usr/bin/cal $m $y
```

```
tigger 212 % date
Wed Jan 28 14:38:38 EST 2009
tigger 213 %
```

Grep

- Prints out all the lines in the input that **matches** an expression
- **grep** [options] **pattern** [file]
- Options let you do inverse search, ignore case, ……
- grep exits with 0 (found) 1 (not fund) 2 (file not found)
- Regular expressions used in grep, sed, vi, awk to match a pattern

Regular Expressions

- "foobar" matches (only) foobar
- '.' Matches any single character
  - f.obar matches faobar, fboar, ….
- [xyz] matches any character in the set
  - fo[abo]bar matches foabar, fobbar, foobar
- [^xyz] matches any character that is **not** in the set
  - fo[^ab]bar matches focbar, fodbar but not foabar

Regular Expressions

- '*' matches 0 or more occurrence of the last char
  - fo* matches f,fo,foo,fooo,foooo
- '?' matches 0 or 1 occurrence of the last char
  - fo?bar matches fbar and fobar
- '+' matches one or more occurrence of the last char
  - fo+bar matches fobar foobar, fooobar, …

Regular Expressions

- '^' matches the beginning of a string
- '$' matches the end of a string
- [a-z] matches any character in the range
- [0-9] matches any digit in the range
  - ^[ABC] matches A,B, or C at the beginning of a string
  - ^[^ABC] matches any character at the beginning of a string except A, B, and C
  - ^[^a-z]$ matches any single character string except a lower case letter

Regular Expressions

- “\<“ and “\>” matches the beginning and end of a word
- \{n\} matches n occurrences of the last char
- \{n,\} at least n occurrences
- \{n,m\} between n and m occurrences
- ^(\+|-)?[0-9]+\.?[0-9]*$ what is that?

```
-123.24 that is a floating point number
786 that is an integer
Regular sentence
Another field
234.23
one sentence with one letter repeated twice in a row
tigger 259 % egrep 'let?er' test
tigger 260 % egrep 'let*er' test
one sentence with one letter repeated twice in a row
tigger 261 % egrep 'let+er' test
one sentence with one letter repeated twice in a row
tigger 262 % egrep 'let?er' test
tigger 263 % egrep 'one s[a-f]' test
one sentence with one letter repeated twice in a row
tigger 264 % egrep '^(\+|-)?[0-9]+\.?[0-9]*$' test
234.23
tigger 265 % egrep '^(\+|-)?[0-9]+\.?[0-9]*' test
-123.24 that is a floating point number
786 that is an integer
234.23
tigger 266 %
```

Other UNIX Utilities

- Uniq sort tr cut find awk (more later) xargs.
- tr x y # replace every occurrence of x by y
- tr ab cd #replace every occurrence of a by c and b by d
- tr “[a-z]” “[A-Z]” <filename
- tr –s a <filename

Other UNIX Utlities

- cut used to split data from files
- cut [**-f**fields] [**-c**column] [**-d**char] filename
- cut **–f**3 **–d**, filename
- cut **–c**30-40 filename
- find . –type d –print
- find –type f –name “*.c” –print // or ‘ ‘
-

Other Unix Utilities

- xargs `commands` execute the given command for each word in its stdin
- find –type f –name *.c –print |xargs wc
- which prog
- whereis prog
- bg and fg
- Command &
- Command; command;

UNIX Commands

- Grouping using () date; who >temp
- (date; who) >temp
- >> file << pattern
- Command && another command
- Command || another command

Run command, if successfule run another command

Quotes

- Escape '\' is used to indicate the next character is not a special character.
- If a file name contains something like '*' data*12, we can refer to it as data*12
- We can use ' ' every character between these two single quotes is treated as non-special except ' `cat 'data*12'`

Quotes

- ` (back-quote) ` the contents of the quote is treated as a shell command
- echo `cat file`
- Double-quote " " like single quote except the variable substitution $ and back-quotes ` are still treated as special characters

Finally

- (command) is executed in a subshell
- B=4
- B=5
- echo $B vs.
- B=4
- (B=5)
- echo $B
- Forking

Set B=5 %% for csh