EECS2031 Software Tools

F 2020-21

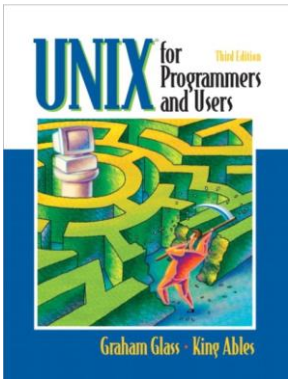
Nov 24/25, 2021 Lecture 21

19

Contents

- Overview of UNIX
  - Structures
  - File systems
    - Pathname: absolute vs relative
    - Security `-rwx-rw--x`
  - Process:
    - Exit code ≥ 0
    - IPC: Pipes

      who | sort who | sort | head -3



Previous lecture

- **Utilities/commands**
  - **Basic: pwd, ls, rmdir, mkdir, cat, more, mv, cp, rm, wc, chmod**
  - Advanced: grep/egrep, sort, find ….
- Shell and shell scripting language

20

20

Unix Utilities

- Standard UNIX comes complete with at least 200 small utility programs, usually including:
  - shells,
  - editors,
  - a C compiler,
  - matching with regular expressions,
  - searching,
  - a sorting utility,
  - software development tools,
  - text-processing tools, etc.

21

21

Basic utilities/commands

Introduced the following utilities, listed in in groups:

General	Directory	File	File print
man	pwd	cat	lp
clear	mkdir -p	more less	lpr
echo	ls -l -d -a -R -S -t -r	head tail	lprm
date	cd	cp -r	lpq
cal	rmdir must be empty	mv move and/or rename	lpstat
		rm -r	
		file	
		wc -l -c -w	
		chmod g+w 750	
		chgrp	
		chown	
		newgrp	

cp directory needs -r
mv directory does not
rm directory needs -r

24

24

```
red 302 % which rm
rm:      aliased to rm -i
red 303 %
```

```
red 303 % alias
cd       cd !* ; setXwd
cp       cp -i
l.       ls -d .* --color=auto
ll       ls -l --color=auto
ls       ls --color=auto
mc       source /usr/libexec/mc/mc-wrapper.csh
module   (set _prompt="$prompt";set prompt="";eval `/usr
/usr/bin/test 0 = $_exit;)
mv       mv -i
popd     popd ; setXwd
pushd    pushd !* ; setXwd
rm       rm -i
setXwd   /cs/local/bin/setXtermTitle "${HOST}:`pwd`"
vi       vim
red 304 %
```

In the login shell (tcsh), as a safeguard,

- when you issue cp, it is replaced by cp -i
- when you issue mv, it is replaced by mv -i
- when you issue rm, it is replaced by rm -i

In other shells, should use -i

cp vs mv

- cp file1 abc cp -r dir abc
  mv file1 abc mv dir1 abc

- Copy (copy+paste) and move (cut+paste) a 3G movie, which is faster?
- Below will both rename file1 to file2, what's the difference?

  cp file1 file2
  rm file1

  mv file1 file2

Counting Lines, Words and Chars in Files: WC

wc -lwc {fileName}*

- The wc utility counts the number of lines, words, and/or characters in a list of files.
- If no files are specified, standard input is used instead.

- -l option requests a line count,
- -w option requests a word count,
- -c option requests a character count.

- If no options are specified, then all three counts are displayed.

- A word is defined by a sequence of characters surrounded by tabs, spaces, or new lines.

27

27

Counting Lines, Words and Characters in Files: wc

- For example, to count lines, words and characters in the "heart.txt" file, we used:

```
$ wc  heart.txt        # or wc  < heart.txt
  9     43     213    heart.txt
```

obtain a count of the number of lines, words, and characters

- Given class list file "EECS2031", in which each line represents one student. How many students are there in the class? Let's do it

```
$ wc -l  EECS2031
$ cat EECS2031  |  wc  -l      # another way, using pipe

$ wc -l EECS2031.LAB01  EECS2031.LAB02   #also get total
```

- How many people are currently logging onto EECS server?

```
$ who | wc -l          # (have to) use pipe
```

29

29

indigo.cse.yorku.ca - PuTTY

```
sh-4.2$ wc -l EECS2031*
   129 EECS2031M
    64 EECS2031M.LAB01
    65 EECS2031M.LAB02
   104 EECS2031N
    58 EECS2031N.LAB01
    28 EECS2031N.LAB02
    18 EECS2031N.LAB03
   122 EECS2031O
    58 EECS2031O.LAB01
    64 EECS2031O.LAB02
   710 total
sh-4.2$ wc -l EECS2031O
122 EECS2031O
sh-4.2$ wc -l EECS2031O*
  122 EECS2031O
   58 EECS2031O.LAB01
   64 EECS2031O.LAB02
  244 total
sh-4.2$ wc -l EECS2031O EECS2031N
  122 EECS2031O
  104 EECS2031N
  226 total
sh-4.2$ wc -l EECS2031?
  129 EECS2031M
  104 EECS2031N
  122 EECS2031O
  355 total
sh-4.2$
```

cat EECS2031* | wc -l

cat EECS2031O | wc -l

cat EECS2031O* | wc -l

cat EECS2031O EECS2031N | wc -l

cat EECS2031? | wc -l

YORK UNIVERSITÉ UNIVERSITY

31

Pipeline Example2

How many users are logged in?

who > tmp.txt wc -l < tmp.txt

Another way

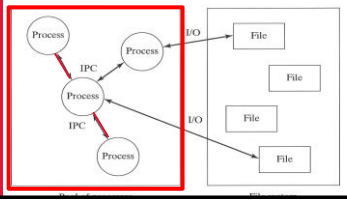
32

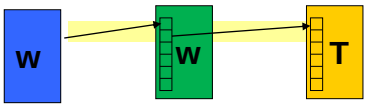
Pipeline Example2

How many users are logged in?

who > tmp.txt wc -l < tmp.txt

who | wc -l





33

of entries in a directory?

indigo.cse.yorku.ca - PuTTY

```
sh-4.2$ ls -l
total 0
drwxrwx--- 2 webapp submit  63 Feb 19 13:01 huseyn
drwxrwx--- 2 webapp submit 126 Feb 19 13:14 khalid22
drwxrwx--- 2 webapp submit  94 Feb 19 12:44 rohit06
drwxrwx--- 2 webapp submit  98 Feb 19 12:48 samishah
drwxrwx--- 2 webapp submit  98 Feb 19 12:57 seenter
drwxrwx--- 2 webapp submit  98 Feb 19 13:05 unaeem
drwxrwx--- 2 webapp submit  98 Feb 19 13:02 wazalif
sh-4.2$ ls -l | wc -l
8
sh-4.2$ ls
huseyn  khalid22  rohit06  samishah  seenter  unaeem  wazalif
sh-4.2$ ls | wc -w
7
sh-4.2$
```

works if entry name does not have spaces

34

Extend lab, check how many submitted

34

File Attributes

- We used ls to obtain a long listing of "heart.txt" and got the following output:

```
$ ls  -l      heart.txt
-rw-r--r--  1  huiwang faculty 213   Jan  31  00:12  heart.txt
$ ls  -ld     lyrics
drwxr-xr-- 1  huiwang faculty 533  Jan  31  10:22  lyrics
$_
```

name of the file

time that the file was last modified

size of the file, in bytes

group of the owner of the file

username of the owner of the file

hard-link count of the file

type and permission mode of the file

File Attributes

- **File Types**
  - first field describes the file's type and permission settings.

drwxr-xr-- 1 huiwang faculty 533 Jan 31 10:22 lyrics

- rw-r--r-- 1 huiwang faculty 213 Jan 31 00:12 heart.txt

- The first character indicates the type of file, which is encoded as follows :

character	File Type
-	regular file
d	directory file
b	buffered special file(such as a disk drive)
c	unbuffered special file(such as a terminal)
l	symbolic link
p	pipe
s	socket

Determining Type of a File: file

file fileName(s)

- The file utility attempts to describe the contents of the fileName argument(s), including the language in which any of the text is written.
- not reliable; it may get confused.

$ file heart.txt # determine the file type.
heart.txt: ASCII text
$ file lab5B.c
lab5B.c: C source, ASCII text
$ file a.out
a.out: ELF 64-bit LSB executable, x86-64, version 1 (SYSV) .....

$ file course.txt ASCII text

38

File Permissions (Security) -- revisit

- File permissions are the basis for file security. They are given in three clusters.

$ ls -l heart.txt
- rw- r-x r-- 1 huiwang faculty 213 Jan 31 00:12 heart.txt

User (owner)	Group	Others
r w -	r - x	r - -

clusters

Each cluster of three letters has the same format:

Read permission	Write permission	Execute permission
r	w	x

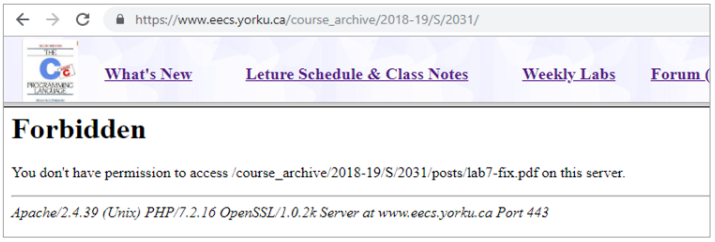
e.g., webfile: others need to have r permission
submit dir: group need to have w permission

39

39

Permission examples

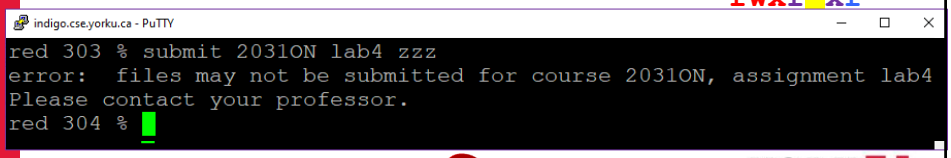
Webfile accessible: others must has r permission



-rwxr-x---

submit directory open: group must has w permission

-rwxr-xr--



40 How to set/change permission?  chmod

YORK UNIVERSITÉ UNIVERSITY

40

Change File Permissions: chmod

Only owner and admin can change

chmod -R change{, change}* {fileName }+

- The **chmod** utility changes the **modes (permissions)** of the specified files according to the change parameters, which may take the following forms:

clusterSelection + newPermissions (add permissions)
clusterSelection - newPermissions (subtract permissions)
clusterSelection = newPermissions (assign permissions absolutely)

where **clusterSelection** is any combination of:

- **u** (user/owner)
- **g** (group)
- **o** (others)
- **a** (all)

rw- r-x r--
u g o
a

newPermissions is any combination of
r (read) **w (write)** **x (execute)**

41 • The **-R** option recursively changes the modes of the files in directories.

41

rw- r-x r--
u g o
a

Changing File Permissions: examples

Requirement	Change parameters
Add group a write permission	chmod g+w file/dir
Remove user (owner) a write permission	u-w
Remove other's read and write permission	o-rw o-wr o-r,o-w
Add execute permission for user, group, and others.	a+x u+x,g+x,o+x ugo+x guo+x
Give the group read permission only.	g=r
Add write permission for user, and remove group read permission.	u+w, g-r
Give the other read and execute permission	o=wx o=xw

overwrite old

overwrite old

YORK UNIVERSITÉ UNIVERSITY

42

Changing File Permission: examples

rw- r-x r--
u g o
a

- Here's an example of how to set these permissions:

```
$ ls -l lab4.pdf                # not accessible on web
 -rw-r-----  1   huiwang    faculty  213  Jan  31  00:12  lab4.pdf
$ chmod o+r lab4.pdf      # accessible now
$ ls -l lab4.pdf
 -rw-r--r--  1   huiwang    faculty  213  Jan  31  00:12  lab4.pdf
$ chmod a+x  lab4.pdf
$ ls -l lab4.pdf
 -rwxr-xr-x  1   huiwang    faculty  213  Jan  31  00:12  lab4.pdf


$ ls  -ld  2031A        # list attributes of directory 2031M itslef.
 drwxr-xr-x   45   huiwang  faculty 4096   Apr   29   14:35
$ chmod   o-rx 2031A          # other more  rx
$ ls  -ld   2031A
 drwxr-x---   45   huiwang  faculty  4096  Apr  29  14:35
$
```

43

Stopped here

43

Changing Permissions Using Numbers chmod 764

- The chmod utility allows you to specify the new permission setting of a file as 3 octal numbers (0~7) .

- Each octal digit (0~7) represents a permission triplet.
  binary 1/0 1/0 1/0
  r w x

For example, if you wanted a file to have the permission settings of

rwx rw- --- # owner: rwx, group: r x → chmod u=rwx, g=rw,o=r

then the octal permission setting would be 764, calculated as follows:

	User	Group	Others
setting	rwx	rw-	r--
binary	111	110	100
octal	7	6	4

44

44

Changing Permissions Using Numbers chmod 750

- The chmod utility allows you to specify the new permission setting of a file as 3 octal numbers (0~7) .

- Each octal digit (0~7) represents a permission triplet.
  binary 1/0 1/0 1/0
  r w x

For example, if you wanted a file to have the permission settings of

rwx r-x --- # owner: rwx, group: r x → chmod u=rwx, g=rx,o=

then the octal permission setting would be 750, calculated as follows:

	User	Group	Others
setting	rwx	r-x	---
binary	111	101	000
octal	7	5	0

45

45

Changing File Permissions Using Octal Numbers

- The octal permission setting would be supplied to chmod as follows:

```
$ chmod  750  lab4.pdf          # or  chmod u=rwx, g=rx,o=    lab4.pdf
$ ls  -l  lab4.pdf              # confirm.
  - rwx r-x ---      45   huiwang faculty    4096    Apr  29 14:35 lab4.pdf
$ _
```

7	5	0
⇩		
111	101	000
⇩		
rwx	r-x	- - -

46

46

Changing Permissions Using Octal Numbers

- The chmod utility allows you to specify the new permission setting of a file as an octal number.

rwx	7	Read, write and execute	111
rw-	6	Read, write	110
r-x	5	Read, and execute	101
r--	4	Read,	100
-wx	3	Write and execute	011
-w-	2	Write	010
--x	1	Execute	001
---	0	no permissions	000

⬇

chmod u=rwx,g=rwx,o=rx	chmod 775
chmod u=rwx,g=rx,o=	chmod 750
chmod u=rw,g=r,o=r	chmod 644
chmod u=rw,g=r,o=	chmod 640
chmod u=rw,go=	chmod 600
chmod u=rwx,go=	chmod 700

chmod 700 sample.out

47

An example: setting up submit directory using chmod

- https://wiki.eecs.yorku.ca/dept/tdb/services:submit:submit-setup

Department of Electrical Engineering & Computer Science

Technical Database

News
Departmental Services
E-Mail
Lab Schedules
Login and Remote Access
Operating System
Policies and Procedures
Printing
Scanning
Software
Web Publishing
Wiki Publishing

SUBMIT DIRECTORY SETUP

Technical Database » Departmental Services » Submit » Submit Directory Setup

In order to setup a submit directory for your course:

- The course directory must be under /eecs/course.
- In the course directory, create a directory called "submit". That should be accessible by everyone.
- Under the submit folder, create one directory per assignment. The assignment directory r must be writable by group, not by "other".

For example, to setup a submit directory for course 1021 and assignment a1, use the following c

```
% mkdir /eecs/course/1021 <- this is only necessary if you haven't creat
% chmod 755 /eecs/course/1021
% mkdir /eecs/course/1021/submit
% chmod 755 /eecs/course/1021/submit
% mkdir /eecs/course/1021/submit/a1
% chgrp submit /eecs/course/1020/submit/a1
% chmod 770 /eecs/course/1021/submit/a1
```

111 111 000
rwx rwx ---

If you no longer wish to allow submissions for an assignment (e.g. past a due date) then remove write permission from the group

```
chmod g-w /eecs/course/1021/submit/a1
```

48

- Also next page Let's do it

48

Example of chmod, chgrp

- Create a submission directory for weekly lab
  - mkdir lab7
  - chgrp submit lab7 # change group to 'submit'
  - chmod 770 lab7 # or chmod u=rwx, g=rwx,o= lab7

$ mkdir lab7
$ ls -ld lab7
drwx------ 2 huiwang **faculty** 4096 Jul 11 16:39 lab7

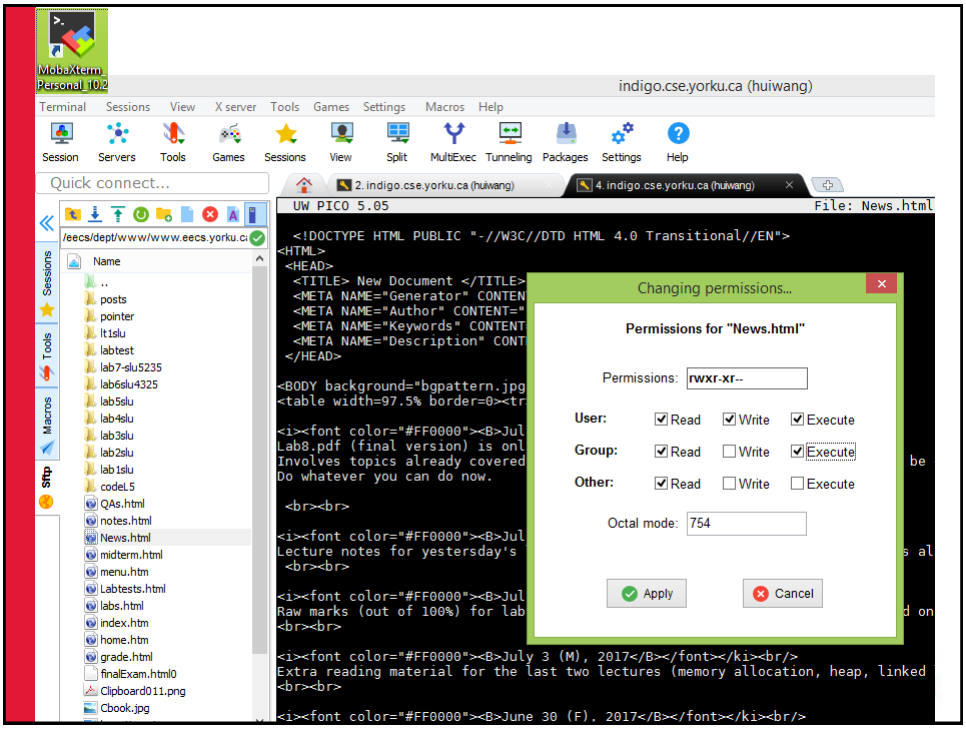
$ chgrp submit lab7
$ ls -ld lab7
drwx------ 2 huiwang **submit** 4096 Jul 11 16:39 lab7

$ chmod 770 lab7
$ ls -ld lab7
drwxrwx--- 2 huiwang submit 4096 Jul 11 16:39 lab7
7 7 0

- After due time, close the submission
  $ chmod g-w lab7 # or chmod 750 lab7
  drwxr-x--- 2 huiwang submit 4096 Jul 11 16:39 lab7

49

49



50

Basic utilities/commands

Introduced the following utilities, listed in in groups:

General
- man
- clear
- echo
- date

Directory
- pwd
- mkdir -p
- ls -l -d -a -R -S -t -r
- cd
- rmdir must be empty

File
- cat
- more less
- head tail
- cp -r
- mv move and/or rename
- rm -r
- wc -l -c -w
- file
- chmod g+w 750
- chgrp
- chown
- newgrp

File print
- lp
- lpr
- lprm
- lpq
- lpstat

cp directory needs -r
mv directory does not
rm directory needs -r

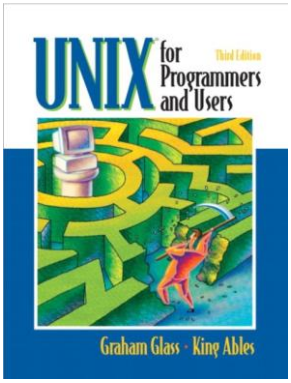
51

51

Contents

- Overview of UNIX
  - Structures
  - File systems
    - Pathname: absolute vs relative
    - Security -rwx-rw--x
  - Process:
    - Exit code ≥ 0
    - IPC: Pipes

      who | sort who | sort | head -3



- **Utilities/commands**
  - Basic: pwd, ls, rmdir, mkdir, cat, more, mv, cp, rm, wc, chmod
  - **Advanced: grep/egrep, sort, cut, find ….**

    today
- Shell and shell scripting language

52

52

Utilities II – advanced utilities

Introduces utilities for power users, grouped into logical sets

We introduce about thirty useful utilities.

Section	Utilities
Filtering files	egrep, fgrep, grep, uniq
Sorting files	sort
Extracting fields	cut
Comparing files	cmp, diff
Archiving files	tar, cpio, dump
Searching for files	find
Scheduling commands	at, cron, crontab
Programmable text processing	awk, perl
Hard and soft links	ln
Switching users	su
Checking for mail	biff
Transforming files	compress, crypt, gunzip, gzip, sed, tr, ul, uncompress
Looking at raw file contents	od
Mounting file systems	mount, umount
Identifying shells	whoami
Document preparation	nroff, spell, style, troff
Timing execution of commands	time

54

54

Filtering Files grep, uniq

- **grep, egrep, fgrep** "Global/Get Regular Expression and Print"

  -w -i -v

  - Filter out all lines that do not contain a specified pattern,
  - Giving you the line that contains the specified pattern

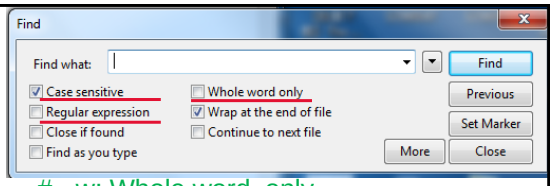
$ cat inputFile.txt # list the file to be filtered
line1 Well you know it's your bedtime,
line2 So turn off the light,
line3 Say all your prayers and then,
line4 Oh you sleepy young heads dream of wonderful things,
line5 Beautiful mermaids will swim through the sea,
line6 And you will be swimming there too.

$ grep the inputFile.txt # search for the word "the"
line2 So turn off **the** light,
line3 Say all your prayers and **the**n,
line5 Beautiful mermaids will swim through **the** sea,
line6 And you will be swimming **the**re too.

55

55

Searching for Regex: grep



$ grep -w the inputFile.txt # -w: Whole word only
line2 So turn off the light, Lines that contain "the" as whole word
line5 Beautiful mermaids will swim through the sea,

$ cat inputFile.txt | grep -w the

$ grep -v -w the inputFile.txt # -v: reverse the filter.
line1 Well you know it's your bedtime, Lines that don't contain "the" as whole word
line3 Say all your prayers and then,
line4 Oh you sleepy young heads dream of wonderful things,
line6 And you will be swimming there too.

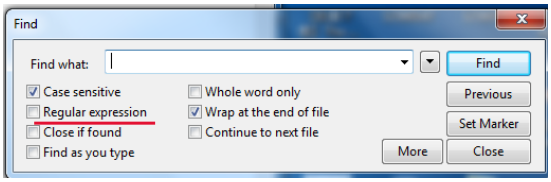
$ grep -i -w the inputFile.txt # ignore case, default case sensitive

$ grep -w Wang classlist # who have family name Wang?
$ grep -w Wang classlist | wc -l # how many ?
$ grep -w Wang classlist > fileWang # write to a file

56

56

Searching for Regex: grep



How to use grep to search lines that contain numbers?

$ grep ? inputFile.txt

How to use grep to search lines that contain lower case letters?

$ grep ? inputFile.txt

Given String s = "abs0deb2afg43affe6wqf53sd5", how to replace all digits in it with character 'X' in Java

Utility	Kind of pattern that may be searched for
fgrep	fixed string only
grep	regular expression
egrep	extended regular expression

Regular Expressions

What is a Regular Expression?

- A regular expression (regex) describes a pattern to match multiple input strings.
- Regular expressions descend from a fundamental concept in Computer Science called finite automata theory

- Regular expressions are endemic to Unix
  - Some utilities/programs that use Regex:
    - **vi**, **ed**, **sed**, and **emacs**
    - **awk**, **tcl**, **perl** and **Python**
    - **grep**, **egrep**
    - **Compilers** **scanf (" %[^\n]s ", str);**

For this course

- The simplest regular expression is a string of literal characters to match.
- The string **matches** the regular expression if it contains the substring.

59

59

Regular Expressions

Exact Matches

Match one any char **.**

Alternate []

[ab]
[^ab]
[a-d]

Anchors ^ $

Repetitions

* 0 or more
? 0 or 1
\+ 1 or more

60

YORK UNIVERSITÉ UNIVERSITY

60

Regular Expressions: Exact Matches

regular expression → cks

$ grep cks inputFile.txt

UNIX Tools rocks.

match

UNIX Tools sucks.

match

UNIX Tools is okay.

no match

61

61

Regular Expressions: Multiple Matches

- A regular expression can match a string in more than one place.

regular expression → apple

$ grep apple inputFile.txt

Scrapple from the apple.

match 1 *match 2*

$ grep -w apple inputFile.txt ?

62

62

Regular Expressions

Exact Matches

Match one any char .

Alternate []

[ab]
[^ab]
[a-d]

Anchors ^ $

Repetitions

* 0 or more
? 0 or 1
+ 1 or more

63

63

Regular Expressions: Matching Any Character

- The . regular expression can be used to match any one character.

regular expression → o.

$ grep o. inputFile.txt

Force me to put on that

match 1 *match 2*

o.c ?

64

$ grep -w o. inputFile.txt ?

64

Regular Expressions

Exact Matches

Match one any char .

Alternate []

[ab]

[^ab]

[a-d]

Anchors ^ $

Repetitions

* 0 or more

? 0 or 1

+ 1 or more

Regular Expressions: Alternate Character Classes

- Character classes `[]` can be used to match any specific set of characters.

regular expression → b [eor] a t

$ grep b[eor]at inputFile.txt

beat a brat on a boat

match 1 *match 2* *match 3*

Does not match beoat

b[eor][or]at will do bat?

- [aeiou] will match any of the characters a, e, i, o, u
- [kK]orn will match korn or Korn

Regular Expressions: Alternate Character Classes

- Character classes `[]` can be used to match any specific set of characters.

regular expression → b [eo] a t

$ grep b[eo]at inputFile.txt

beat a brat on a boat

match 1 *no match* *match 2*

Does not match beoat
b[eor][or]at will do
bat?

- [aeiou] will match any of the characters a, e, i, o, u
- [kK]orn will match korn or Korn

67

67

Regular Expressions: Negated Character Classes

- Character classes can be negated with the `[^]` syntax.
  - Negate all in []

regular expression → b [^eo] a t

$ grep b[^eo]at inputFile.txt

beat a brat on a boat

no match *match* *no match*

scanf (" %[^\n]s", str);

68

68

Regular Expressions: Other Character Classes

- Other examples of character classes:
  - [0123456789] will match any digit
  - [abcde] will match a b c d e

- Ranges can also be specified in character classes

  [0-9] is the same as [0123456789] $ grep [0-9] inputFile.txt
  [a-e] is equivalent to [abcde]

- You can also combine multiple ranges

  [abcde123456789] is equivalent to [a-e1-9]
  [a-zA-Z] all the letters

  [aeiou] ??

69

Regular Expressions: Named Character Classes

- Commonly used character classes can be referred to by name
  - alpha,
  - lower,
  - upper,
  - alnum,
  - digit,
  - punct,
  - cntl

For your information

- Syntax [[:name:]]
  - [0-9] [[:digit:]] $ grep [[:digit:]] inputFile
  - [a-zA-Z] [[:alpha:]]
  - [a-zA-Z0-9] [[:alnum:]]
  - [45a-z] [45[:lower:]]

- Important for portability across languages

70

70

Regular Expressions

Exact Matches

Match one any char .

Alternate []

[ab]
[^ab]
[a-d]

Anchors ^ $

Repetitions

* 0 or more
? 0 or 1
+ 1 or more

71

71

Regular Expressions: Anchors

- Anchors are used to match at the beginning or end of a line (or both).
  - ^ means beginning of the line — ^ the begin with "the"
  - $ means end of the line — the$ end with "the"

regular expression → ^ b [eor] a t

$ grep ^b[eor]at inputFile.txt

This is brat on a boat ?

beat a brat on a boat

match

regular expression → b [eor] a t $

$ grep b[eor]at$ inputFile.txt

A brat on a boat, rght ?

beat a brat on a boat

match

$

72

Regular Expressions: Anchors

- Anchors are used to match at the beginning or end of a line (or both).
  - ^ means beginning of the line — ^ the begin with "the"
  - $ means end of the line — the$ end with "the"

$grep cse classlist

```
red 32 % grep cse classlist
cse***          *********       Yu      Ying
cse***          *********       Wong    JunXiu
ttCTV*          cse******       Tong    Tracy
red 33 %
```

$grep ^cse classlist

```
red 33 % grep ^cse classlist
cse***          *********       Yu      Ying
cse***          *********       Wong    JunXiu
red 34 %
```

Regular Expressions

Exact Matches

Match one any char .

Alternate []

[ab]
[^ab]
[a-d]

Anchors ^ $

Repetitions

- * 0 or more
- ? 0 or 1
- + 1 or more

Regular Expression: Repetitions

"Kleene Star"

- The * is used to define **zero or more** occurrences of the *single* regular expression preceding it.

regular expression → y a * z

$ grep ya*z inputFile.txt

I got mail, yaaaaaaaaaaz!

match

zero or more occurrences of 'a' (between y z)
yz yaz yaaz yaaaz

$ grep oa*o inputFile.txt

regular expression → o a * o

For me to lbo k on. Take oaao

match *match*

zero or more occurrences of 'a' (between o o)
oo oao oaao oaaao

Regular Expressions: Repetition Ranges, Subexpressions

- Ranges can also be specified
  - `{n,m}` notation can specify a range of repetitions for the immediately preceding regex
  - `{n}` means exactly *n* occurrences
  - `{n,}` means at least *n* occurrences
  - `{n,m}` means at least *n* occurrences but no more than *m* occurrences

- Example:
  `.{0,}` same as `.*`
  `a{2,}` same as `aaa*` **# at least 2 occurrences**
  `a{2}` same as `aa` **# exact 2 occurrences**

For your information

- If you want to group part of an expression so that `*` applies to more than just the previous character, use `( )` notation

- Subexpresssions are treated like a single character
  `a*` matches zero or more occurrences of `a`
  `abc*` matches `ab`, `abc`, `abcc`, `abccc`, … **# ab followed by 0 or more c**
  `a(bc)*` matches `a`, `abc`, `abcbc`, `abcbcbc`, …
  `(abc)*` matches `abc`, `abcabc`, `abcabcabc`, …

Extended Regular Expressions: Repetition Shorthands

- The * (star) has already been seen to specify zero or more occurrences of the immediately preceding character
  - abc*d will match abd, abcd, abccd, or abcccccd

- The + (plus) means one or more occurrence of the preceding character
  - abc+d will match abcd, abccd, or abcccccd
    but will not match abd ✘ one or more occurrence of c

- The ? (question mark) specifies an optional character, the single character that immediately precedes it
  - July? will match Jul or July zero or one occurrence of y
    - Equivalent to (Jul|July)
  - abc?d will match abd and abcd
    but will not match abccd ✘

79

79

Repetition * ? + summary

Regex	Meaning
a*	0 or more a
a?	0 or 1 a
a+	1 or more a

ab*c matches

ab?c matches

ab+c matches

Which infinite or finite?

80

Repetition * ? + summary

Regex	Meaning
a*	0 or more a
a?	0 or 1 a
a+	1 or more a

ab*c matches ac abc abbc abbbc abbbbc ….
ab?c matches ac abc
ab+c matches abc abbc abbbc abbbbc ….

- Don't get confused with filename wildcard *

ls ba* ba followed by 0 or more any char -- anything
ls a*.c a followed by 0 or more any char – anything, then .c

81

(Extended) Regular Expression Summary

Pattern	Maning	Example
c	Non-special, matches itself	'tom'
^	Start of line	'^ab'
$	End of line	'ab$'
.	Any single character	'.nodes'
[…]	Any single character in []	'[tT]he'
[^…]	Any single character not in []	'[^tT]he'
R*	Zero or more occurrences of R	'e*'
R?	Zero or one occurrences of R (egrep)	'e?'
R+	One or more occurrences of R (egrep)	'e+'
R1R2	R1 followed by R2	'[st][fe]'
R1|R2	R1 or R2 (egrep)	'the|The'

anchored

repetition

82

82

(Extended) Regular Expression Summary

Pattern	Maning	Example
c	Non-special, matches itself	'tom'
^	Start of line	'^ab'
$	End of line	'ab$'
.	Any single character	'.nodes'
[…]	Any single character in []	'[tT]he'
[^…]	Any single character not in []	'[^tT]he'
R*	Zero or more occurrences of R	'e*'
R?	Zero or one occurrences of R (egrep)	'e?'
R+	One or more occurrences of R (egrep)	'e+'
R1R2	R1 followed by R2	'[st][fe]'
R1|R2	R1 or R2 (egrep)	'the|The'

Don't get confused

anchored

repetition

83

83

(Extended) Regular Expression Summary

Pattern	Maning	Example
c	Non-special, matches itself	'tom'
^	Start of line	'^ab'
$	End of line	'ab$'
.	Any single character	'.nodes'
[…]	Any single character in []	'[tT]he'
[^…]	Any single character not in []	'[^tT]he'
R*	Zero or more occurrences of R	'e*'
R?	Zero or one occurrences of R (egrep)	'e?'
R+	One or more occurrences of R (egrep)	'e+'
R1R2	R1 followed by R2	'[st][fe]'
R1|R2	R1 or R2 (egrep)	'the|The'

Don't get confused

anchored

repetition

Don't get confused with UNIX metacharacter (filename wildcards)

ls file*.c *.java
cp xFile?.c . one any

84

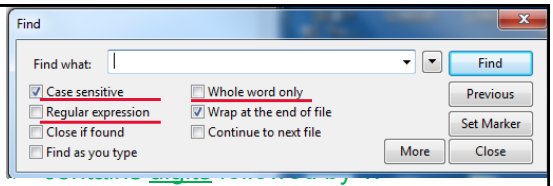
84

Utility	Kind of pattern that may be searched for
fgrep	fixed string only
grep	regular expression
egrep	extended regular expression

- Regular expression and extended expression maybe confusing.
- grep may behave differently in different shells.

- So for this course
  - Use grep -E or egrep take extended Regular Expression
  - Work on Bourne shell (sh) or Bourne again shell (bash)

Examples of Regex, grep



$ grep [0-9]x inputFile.txt

$ grep ^[tT]he inputFile.txt # begins with *the* or *The*

$ grep ^[a-z] inputFile.txt # begins with a lower case letter

$ grep .nd inputFile.txt # contains one any character followed by nd

$ grep [ab]nd$ inputFile.txt # ends with 'and' or 'bnd'

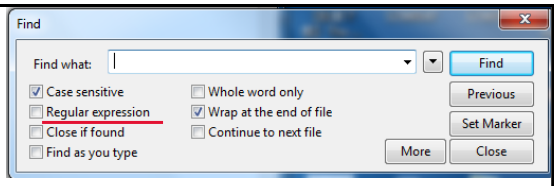
$ grep -w W[ao]ng classlist # who have family name Wang or Wong?

$ grep -w Ch[ae]n classlist | wc -l # how many Chan or Chen?

$ ls -l | grep webapp # who submitted using web submission?

$ who | grep jeff who | grep rogers Gu Guo gu[o] ?

The early question revisit



How to use grep to search lines that contain numbers?

$ grep [0-9] inputFile.txt # or grep [[:digit:]] inputFile.txt

How to use grep to search lines that contain lower case letters?

$ grep [a-z] inputFile.txt # or grep [[:lower:]] inputFile.txt

Given String s= "abs0deb2afg43affe6wqf53sd5", how to replace all digits with character 'X' in Java?

```
s = s.replaceAll("[0-9]", "X");
```

Remove?

replaceAll

```
public String replaceAll(String regex,
                         String replacement)
```

Replaces each substring of this string that matches the given regular expression with the given replacement.

87

87

Exit code of grep/egrep

Matching found: 0 No matching: 1 No such file: 2

```
$ grep Wang classlist
$ echo $?          # display its exit value.
0                  # indicates success.

$ grep Leung classlist
$ echo $?
1                  # indicates failure (not matching).

$ grep Wang  classlistXXX
grep: classlistXXX: No such file or directory
$ echo $?
2                  # indicates failure (not such a file).
```

Look for man
man grep | grep -w "exit"

Used in scripting

88