Shell Programming Tools

Outline

- Regular Expression
- Command: cut
- Command: paste
- Command: sed
- Command: tr
- Command: grep
- Command: sort
- Command: uniq

Regular Expressions

- Regular Expressions are used by several different Unix commands, including ed, sed, awk, grep and, to a limited extent, vi

- They provide a convenient and consistent way of specifying *patterns* to be matched.

- The shell recognizes a limited form of regular expressions when using filename substitution.

(Examples next slides)

Match Any Character (1)

1. Matching any Character: Period (.)

   - A period in a regular expression matches any single character, no matter what it is. The expression, ***r.*** specifies a pattern that matches an ***r*** followed by any single character.
   - The regular expression, ***.x.*** matches an x that is surrounded by any two characters, not necessarily the same.
   - The ed command, ***/ ... /*** searches forward in the file you are editing for the first line that contains any three characters surrounded by blanks.

   (The next slide contains further examples. Assume that the filename *intro* contains the text describing the history of UNIX as printed below.)

Match Any Character (2)

Example:

```
uisacad:/home/user> ed intro
245
1,$p                                       Print all lines
The UNIX operating system was pioneered by Ken
Thomson and Dennis Ritchie at Bell Laboratories
in the late 1960s. One of the primary goals in
the design of the UNIX system was to create an
environment that promoted efficient program
development.
/ ... /                                    Looks for three characters surrounded by blanks
The UNIX operating system was pioneered by Ken
/                                          Repeat last search, continue search
Thomson and Dennis Ritchie at Bell Laboratories
1,$s/p.o/XXX/g                             Change all p.o to XXX
1,$p                                       The result would be (take note on the words in red):
The UNIX operating system was XXXneered by Ken
Thomson and Dennis Ritchie at Bell Laboratories
in the late 1960s. One of the primary goals in
the design of the UNIX system was to create an
environment that XXXmoted efficient XXXgram
development.
```

Match the Beginning of the Line

2. Matching the Beginning of the Line: The Caret (^)

When the caret character **^** is used as the first character in a regular expression, it matches the beginning of the line. The expression **^George** matches the character George only if they occur at the beginning of the line.

Example:

```
uisacad:/home/user> ed intro
245
/^the/                                   Find the line that starts with the
the design of the UNIX system was to create an
1,$s/^/>>/                               Insert >> at the beginning of each line
1,$p
>>The UNIX operating system was pioneered by Ken
>>Thomson and Dennis Ritchie at Bell Laboratories
>>in the late 1960s. One of the primary goals in
>>the design of the UNIX system was to create an
>>environment that promoted efficient program
>>development.
```

Match the End of the Line (1)

3. Matching the End of the Line: The Dollar Sign ($)

The dollar sign $ is used to match the end of the line. The expression **$contents** matches the characters contents only if they are the last characters on the line.

.$

\.$ -> matches any line that ends in a period

^\. -> matches any line that starts with one (good for searching for nroff commands in your text).

Match the End of the Line (2)

Example:

```
uisacad:/home/user> ed intro
245
/\.$/                          Search for a line that ends with a period
development.
1,$s/$/>>/                     Add >> to the end of each line
1,$p                           Prints
The UNIX operating system was pioneered by Ken>>
Thomson and Dennis Ritchie at Bell Laboratories>>
in the late 1960s. One of the primary goals in>>
the design of the UNIX system was to create an>>
environment that promoted efficient program>>
development.>>
1,$s/..$//                     Delete the last two characters from each line
1,$p
The UNIX operating system was pioneered by Ken
Thomson and Dennis Ritchie at Bell Laboratories
in the late 1960s. One of the primary goals in
the design of the UNIX system was to create an
environment that promoted efficient program
development.
```

Match a Choice of Characters: The […] Construct

4. Matching a Choice of characters: The**[…]** Construct

/the/ -> this causes ed to search forward in its buffer until it finds a line containing the indicated string of characters

/[tT]he/ -> this matches a lower or uppercase ***t*** followed immediately by the character ***he***

1,$s/[aeiouAEIOU]//g -> deletes all vowels

/[0-9]/ -> find a line containing a digit

/[^A-Z]/ -> find a line that starts with an uppercase letter

1,$s/[A-Z]/*/g -> change all uppercase letters to *

Match Zero or More Characters (1)

5. Matching Zero or More Characters: The Asterisk (*)
- The * is used by the shell in filename substitution to match zero or more characters. In forming regular expressions, the asterisk is used to match zero or more occurences of the preceding character in the regular expression (which may itself be another regular expression).

X* -> matches zero, one, two, three, * capital X's.

XX* -> matches one or more capital X's because the expression specifies a single X followed by zero or more X's.

.* -> specifies zero or more occurrences of any characters.

e.*e -> (combination of . and *), matches all the characters from the first e on a line to the last one.

[A-Za-z][A-Za-z]* -> matches any alphabetic character followed by zero or more alphabetic characters

[A-Za-z0-9][A-Za-z0-9]* -> matches any alphanumeric character followed by zero or more alphanumeric characters

Match Zero or More Characters (2)

Examples:

```
uisacad:/home/user> ed intro
245
1,$s/e.*e/+++/
1,$p
Th+++n
Thomson and D+++s
in th+++ primary goals in
th+++ an
+++nt program
d+++nt.
========================================
 uisacad:/home/user> ed intro
245
1,$s/[A-Za-z][A-Za-z]*/X/g
1,$p
X X X X X X X X
X X X X X X X
X X X 1960X. X X X X X X
X X X X X X X X X X
X X X X X
X.
```

matches all the characters from the first e on a line to the last one.

matches any alphabetic character followed by zero or more alphabetic characters and replaces with X

Match a Precise Number of Characters (1)

6. Matching a Precise Number of Characters: **\{…\}**

Construct:
\{min, max\} -> min specifies the minimum number of occurrences of the preceding regular expression to be matched, and max specifies the maximum.

Examples:
X\{1,10\} -> matches from one to ten consecutive X's.

[A-Za-z]\{4,7\} -> matches a sequence of alphabetic letters from four to seven characters long.

If only one number is enclosed between the braces, for example: **\{10}**, must be matched exactly that many times.

[a-zA-Z]\{7\} -> matches exactly seven alphabetic characters

.\{10} -> matches exactly ten characters (no matter what they are):

Match a Precise Number of Characters (2)

Example:

```
 uisacad:/home/user> ed intro
245
1,$s/^.\{10\}//                    Delete the first 10 chars from each line
1,$p
perating system was pioneered by Ken
d Dennis Ritchie at Bell Laboratories
e 1960s. One of the primary goals in
of the UNIX system was to create an
t that promoted efficient program
t.

?
1,$s/.\{5\}$//                     Delete the last 5 chars from each line
1,$p
perating system was pioneered b
d Dennis Ritchie at Bell Laborat
e 1960s. One of the primary goa
 of the UNIX system was to crea
t that promoted efficient pr
t.                                 didn't have five characters, match failed, and thus left alone
```

Saving Matched Characters : \(…\) (1)

7. Saving Matched Characters: **\{…\}**

Uses “registers” numbered 1 through 9 to store captured characters matched within a regular expression.

^\(.\) -> matches the first character on the line, whatever it is, and stores it into register 1.

^\(\.\)\1 -> matches the first character on the line and stores it in register 1. Then, the expression matches whatever is stored in register 1, as specified by the \1.

^\(…\)\(…\) -> matches some text the first three characters on the line will be stored into register 1, and the next three characters into register 2.

If only one number is enclosed between the braces, for example: **\{10}**, must be matched exactly that many times.

[a-zA-Z]\{7\} -> matches exactly seven alphabetic characters

.\{10} -> matches exactly ten characters (no matter what they are):

Saving Matched Characters : \(…\) (2)

```
Example:
uisacad:/home/user> ed phonebook
160
1,$p                                     Prints the content
Alice Chebba    973-555-2015
Barbara Swingle 201-555-9257
Liz Stachiw     212-555-2298
Susan Goldberg  201-555-7776
Susan Topple    212-555-4932
Tony Iannino    973-555-1295
1,$s/\(.*\)     \(.*\)/\2 \1/ Switch the two fields, use [tab] to separate
1,$p
973-555-2015 Alice Chebba
201-555-9257 Barbara Swingle
212-555-2298 Liz Stachiw
201-555-7776 Susan Goldberg
212-555-4932 Susan Topple
973-555-1295 Tony Iannino
```

Saving Matched Characters : \(…\) (3)

Let’s break down the expression:
The names and the phone numbers are separated from each other in the phonebook file by a single tab character. The regular expression

\(.*\)   \(.*\)

says to match all the characters up to the first tab and assign them to register 1, and to match all the characters that follow the tab character and assign them to register 2. The replacement string

\2 \1

specifies the contents of register 2, followed by a space, followed by the contents of register 1.

Outline

- Regular Expression
- Command: cut
- Command: paste
- Command: sed
- Command: tr
- Command: grep
- Command: sort
- Command: uniq

Command: cut (1)

This command extracts (that is “cut out”) various fields of data from a data file or the output of a command. The general format of the cut command is:

cut -c*chars file*

where chars specifies what characters you want to extract from each line of *file*. This can consist of:

1. a single number, as in -c5 to extract character 5;
2. a comma-separated list of numbers, as in -c1,15,50 to extract characters 1, 13, and 50; or
3. a dash separated range of numbers, as in -c20-50 to extract characters 20 through 50, inclusive.

To extract characters to the end of the line, omit the second number of the range; so

cut -c5- data -> extracts characters 5 through the end of the line from each line of data and writes the results to standard output.

If file is not specified, cut reads its input from standard input, meaning that ou can use cut as a filter in a pipeline.

Command: cut (2)

Examples:

```
uisacad:/home/user> who                          Display current users
rsalv1   pts/2       Aug 17 16:03
test1s   pts/3       Aug 17 17:40
atest2   pts/3       Aug 17 17:40

uisacad:/home/user> who | cut -c1-8              Display the first 8 characters
rsalv1
test1s
atest2

uisacad:/home/user> who | cut -c1-8 | sort       Display first 8 characters and sort
atest2
rsalv1
test1s
```

Command: cut -d and -f Option

The -d and -f options are used with cut when you have data that is delimited by a particular character. -d defines the delimiting character and –f defines which fields you wish to cut.

Format:

cut -d*dachr* -f*fields file*

Example:

```
uisacad:/home/user> cut -d: -f1 /etc/passwd   Extracts field 1
root
daemon
bin
sys
adm
lp
uucp
nuucp
```

*

Command: cut -d and -f Option

Example:

```
uisacad:/home/user> cut -d: -f1,6 /etc/passwd
root:/
daemon:/
bin:/usr/bin
sys:/
adm:/var/adm
lp:/usr/spool/lp
uucp:/usr/lib/uucp
nuucp:/var/spool/uucppublic
```

Extracts fields 1 and 6

*

Outline

- Regular Expression
- Command: cut
- Command: paste
- Command: sed
- Command: tr
- Command: grep
- Command: sort
- Command: uniq

Command: paste (1)

This command puts (that is “pasted”) various fields of data from a data file or the output of a command. The general format of the cut command is:

paste *files*

where corresponding lines from each of the specified files are pasted together to form single lines that are then written to standard output.

Example:

```
 uisacad:/home/user> cat names
Charlie
Emanuel
uisacad:/home/user> cat numbers
(307) 555-5356
(212) 555-3456

uisacad:/home/user> paste names numbers
Charlie (307) 555-5356
Emanuel (212) 555-3456
```

Command: paste -d Option

The -d option allows the fields to be separated by other characters other than tab character.

Format:
-d*chars*

where ***chars*** is one or more characters that will be used to separate the lines pasted together.

Example:
```
uisacad:/home/user> paste -d':' names numbers
Charlie:(307) 555-5356
Emanuel:(212) 555-3456
```

Command: paste -s Option

The -s option tells paste to paste together lines from the same file, not from alternate files. If one file is specified, the effect is to merge all the lines from the file together, separated by tabs, or by the delimiter characters specified with the -d option.

Example:

```
 uisacad:/home/user> cat names
Charlie
Emanuel

 uisacad:/home/user> paste -s names
Charlie Emanuel
```

Outline

- Regular Expression
- Command: cut
- Command: paste
- Command: sed
- Command: tr
- Command: grep
- Command: sort
- Command: uniq

Command: sed

sed stands for stream editor. It is a program used for editing data.

Format:

sed *command file*

where command is an ed-style command applied to each line of the specified *file*. If no file is specified, standard input is assumed.

Examples:

```
uisacad:/home/user> sed -n '1,2p' intro
The Unix operating system was pioneered by Ken
Thomson and Dennis Ritchie at Bell Laboratories
```

Substitute Unix with UNIX

```
uisacad:/home/user> sed 's/Unix/UNIX/' intro
The UNIX operating system was pioneered by Ken
Thomson and Dennis Ritchie at Bell Laboratories
in the late 1960s. One of the primary goals in
the design of the UNIX system was to create an
environment that promoted efficient program
development.
```

Command: sed -n Option

The -n Option
- This option tells sed to extract some lines from a file.

Examples:

```
uisacad:/home/user> sed -n '1,2p' intro
The UNIX operating system was pioneered by Ken
Thomson and Dennis Ritchie at Bell Laboratories
```

Print the first 2 lines

```
uisacad:/home/user> sed -n '/UNIX/p' intro
The UNIX operating system was pioneered by Ken
the design of the UNIX system was to create an
```

Just print lines containing UNIX

Command: sed - deleting lines

To delete entire lines of text, use the d command. By specifying a line number or range of numbers, you can delete specific lines from the input.

Examples:

```
uisacad:/home/user> sed '1,2d' intro
in the late 1960s. One of the primary goals in
the design of the UNIX system was to create an
environment that promoted efficient program
development.
```

Delete lines 1 and 2

```
uisacad:/home/user> sed '/UNIX/d' intro
Thomson and Dennis Ritchie at Bell Laboratories
in the late 1960s. One of the primary goals in
environment that promoted efficient program
development.
```

Delete all lines containing UNIX

Outline

- Regular Expression
- Command: cut
- Command: paste
- Command: sed
- Command: tr
- Command: grep
- Command: sort
- Command: uniq

Command: tr

The **tr** filter is used to translate one character into another.

Format:

tr *from-chars to-chars*

where *from-chars* and to -chars are one or more single characters. Any character in *from-chars* encountered on the input will be translated into the corresponding character in *to-chars.* The result of the translation is written to standard output.

Example:

```
uisacad:/home/user> tr e x < intro
Thx UNIX opxrating systxm was pionxxrxd by Kxn
Thomson and Dxnnis Ritchix at Bxll Laboratorixs
in thx latx 1960s. Onx of thx primary goals in
thx dxsign of thx UNIX systxm was to crxatx an
xnvironmxnt that promotxd xfficixnt program
dxvxlopmxnt.
```

*Translate **e** into **x***

Command: tr -s Option

The -s Option
- This option is used to “squeeze” out multiple occurrences of characters in t-chars.

Examples:

```
uisacad:/home/user> cat lotsaspaces
This        is  an   example    of   a
file    that    contains  a     lot
of        blank    spaces.

uisacad:/home/user> tr -s ' ' ' ' <
lotsaspaces
This is an example of a
file that contains a lot
of blank spaces.
```

Command: tr - d Option

To delete single characters from a stream of input.

Format:
tr -d *from-chars*

Examples:

```
uisacad:/home/user> tr -d ' ' < intro
TheUNIXoperatingsystemwaspioneeredbyKen
ThomsonandDennisRitchieatBellLaboratories
inthelate1960s.Oneoftheprimarygoalsin
thedesignoftheUNIXsystemwastocreatean
environmentthatpromotedefficientprogram
development.
```

Note:
The spaces are deleted.

Outline

- Regular Expression
- Command: cut
- Command: paste
- Command: sed
- Command: tr
- Command: grep
- Command: sort
- Command: uniq

Command: grep

grep allows searching one or more files for particular character patterns.

Format:

grep *pattern files*

Examples:

```
uisacad:/home/user> grep UNIX intro
The UNIX operating system was pioneered by Ken
the design of the UNIX system was to create an
```

```
uisacad:/home/user> grep '[tT]he' intro
The UNIX operating system was pioneered by Ken
in the late 1960s. One of the primary goals in
the design of the UNIX system was to create an
```

Command: grep - Examples and Options

grep - Examples

Command	Prints
grep ‘[A-Z]’ list	Lines from list containing a capital letter
grep ‘[0-9]’ data	Lines from data containing a number
grep ‘[A-Z]...[0-9]’ list	Lines from list containing five-character patterns that start with a capital letter and end with a digit
grep ‘\.pic$’ filelist	Lines from filelist that end in .pic

grep - Options

Option	Example	Description
-v	grep -v ‘UNIX’ intro	Prints all lines that don’t contain UNIX
-l	grep ‘Move_history’ *.c	Find Move_history in all C source files in the current directory
-n	grep -n ‘Move_history’ testch.c	Precede matches with line number

Outline

- Regular Expression
- Command: cut
- Command: paste
- Command: sed
- Command: tr
- Command: grep
- Command: sort
- Command: uniq

Command: sort

sort takes each line of the specified input file and sorts it into ascending order. Special characters are sorted according to the internal encoding of the characters.

Format:

Sort *file*

Example:

```
 uisacad:/home/user> sort phonebook
Alice Chebba     973-555-2015
Barbara Swingle 201-555-9257
Liz Stachiw      212-555-2298
Susan Goldberg  201-555-7776
Susan Topple     212-555-4932
Tony Iannino     973-555-1295
```

Note: The textbook describes the option +1n to skip a field for sorting; however, uisacad5 uses the option –*k n* Example: **sort –k 3** will use the third field of data as the sort key

Command: sort - Examples and Options

sort - Options

Option	Example	Description
-u	sort -u names	Eliminate duplicate lines from the ouput
-r	sort -r names	Reverse the order of the sort
-o	sort names -o sorted_names	Specify the output file
-n	sort -n data	Sort arithmetically
-k n	sort-k 3 data	Use the third field as the sort key
-t	sort +2n -t: /etc/passwd	Skip first two field and sort on the third field.

Outline

- Regular Expression
- Command: cut
- Command: paste
- Command: sed
- Command: tr
- Command: grep
- Command: sort
- Command: uniq

Command: uniq

uniq command is used to find duplicate lines in a file.

Format:

uniq *in_file out_file*

uniq copies *in_file* to *out_file,* removing any duplicate lines in the process. uniq's definition of duplicate lines are consecutive-occurring lines that match exactly.
If *out_file* is not specified, the result will be written to standard output.

Examples and Options:

Example	Description
sort names | uniq	Sort then filters out duplicates
sort names | uniq -d	List duplicate lines
sort /etc/passwd | uniq -d	Find duplicate entries in /etc/passwd
sort /etc/passwd | cut -f1 -d: | uniq -d	Find duplicates
sed '166d' /etc/passwd > /tmp/passwd	Remove the entry 166 and save to /tmp/passwd
sort names | uniq -c	Count line occurences

Reference

1. Unix Shell Programming, Kochan and Wood, 3rd Ed.: Chapter 4