Operating systems 1, Lecture 4

Sanda-Maria AVRAM, Ph. D.

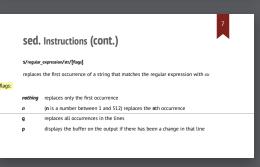
awk

Unix filters Any command that reads a file from the standard input converts it in some way and displays it at the standard output. sed uniq grep

sed. Conditions true for all lines in the file true for the n-th line (lines are numbered cumulatively in the file list) true condition for the last line in the file



sed. Instructions displays the temporary buffer at the standard output delete the temporary buffer which it displays at the standard output before the processed line analogue to i\ but displays the text after processing each line (where str1 and str2 have equal lengths) performs a translation y/str1/str2/ by replacing the characters in the input files found in str1 with the corresponding characters in str2







Extended regular expression (ERE) (cont.)			
{n}	(where n is a number between 0 and 255)		
ful.	repeat the previous expression of exactly n times		
{n,}	repeats the previous regular expression at least n times		
	repeats the previous regular expression of at least ${\bf n}$ times		
{n,m}	and at most m times		
(regular_exp)	group several characters into an expression		
\n	replaces a string with the nth regular expression		
/n	found in the brackets ()		
regexp1 regexp2	matches either regexp1 or regexp2		

echo abcdl	23ef sed -E 's/([a-z]*).*/\l/' # displays only the first group of letter
# intercha	nge the first two words:
echo abcd	efg sed =E 's/([a-z]*) ([a-z]*)/\2 \1/'
# eliminat	es the space between words:
echo abcd	efg sed =E 's/([a-z]*) ([a-z]*)/\1\2 /'
# white sp	ace - space, TAB
sed -E 's/	^[\t]*//g' fl # removes all white spaces at the beginning of the lines
sed -E 's/	[\t]*\$//g' fl # eliminates all white spaces at the end of the lines

awk		
Processes text files by selecting	those lines that satisfy the conditions imposed by a list of templates	(regular
expressions). Its name comes fi	om its three designers and implementers: A. Aho, P. Wieinberger and	B. Kernir
awk [-f script_file] [-F	c] [script] [-V variable=value] [file_list]	

awk. Condition

FS

ORS

ARGC

- is a logical exp	ression built with C operators: , &&, !, (). Operands can be arithmetic expressions, relational
expressions, cons	tants and variables. Variables must not be declared, they are automatically initialized, their type
deducted from ti	e context. For strings there is the concatenation operator (space) as well as some string functions.
Arrays can be us	ed, whose indices can be numerical values or strings.
	Predefined conditions:
BEGIN	it is true before the first line of the first file
END	it is true after the last line of the last file
awk. P	redefined variables
NF	(Number of Fields) the number of words/fields in the current line
NR	(Number of Records) the number of the current processed line
	(the countdown starts at 1): line 1 is the first line of the first file
	the contact in state at 1), the 1 is the first the or the first the

(Field Separator) word/field separator the name of the current file being processed

record separator at output (default is new line) string of command line parameters

the number of command line parameters

sed (stream editor = noninteractive text editor) sed [-n] [-e script] [-f script_file] [file_list] script: condition instruction

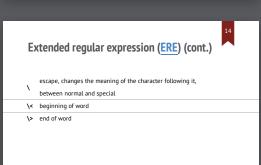
sed. Conditions. Examples	
sed 1,10 instruction file1	
# executes instructions on lines from 1 to 10	
sed 10,8 instruction file1 file2	
# executes instruction on lines from 10 to the end of the file # obtained by conclating file1 with file2	

sed p fil	el # displays each line twice	
sed -n p	filel # displays each line once	
# script_		
	1\	
	11111	
sed -f sc	ript_file filel	

	. Instructions (cont.) Examples
echo	Sunnight # ==> Sunnight
# if	one wants to replace /usr/local/bin with /common/bin
sed '	s/\/usr\/local\/bin/\/common\/bin/' old > new
sed '	a_/usr/local/bin_/common/bin_' old > new
sed '	/usr/local/bin / common/bin ' old > new
sed '	:/usr/local/bin:/common/bin:' old > new
sed /	RES/s/yesyes/noway/g old # replace only in lines containing YES
sed -	'/baz/s/foo/bar/p' old # displays only lines that have changed, due to

gro	ep. Options
-c	(count) displays only the number of lines that match the regular expression
-h	(hide) does not display the file name
-i	(ignore case) does not make the difference between upper and lower case letters
-l	only displays the file names that contain the string that is searched
-n	displays the lines that match the regular expression preceded by the line number relative to the beginning of each file
-q, -s	displays nothing, to determine whether or not there was at least one match
-v	displays lines that do not contain the given string
-w	displays the lines where the string you are looking for is an entire word
-e	is used if we want the regular expression to begin with **

	Exte	ended regular expression (ERE)
	^	the beginning of the line (if ^ is the first character in the regular expression)
!	\$	the end of the line (if \$ is the last character in the regular expression)
		any character
	[list]	any character in list
- 1	[c1-c2]	any character between c1 and c2 in lexicographic order
	[^list]	the negation of [list]
'	•	repeats the previous regular expression as many times as possible
	+	as *, but repeats once or more
1	?	as *, but repeats once or zero times



gre	p. Examples with regular expressions
grep '	Ab file1 file2 # displays the lines in file1 and file2 starting with Ab
grep :	dS filel file2 # displays the lines in file1 and file2 ending with nd
	E '^(.*) (.*) \1\$' file1 \$ displays all lines that begin and end with the ord, separator being space, and contain more than two words
grep -	E 'o(3)' old # displays lines containing "coo"
grep -	E 'o(2,)' old # displays lines containing "oo", or "ooo",
grep -	E 'fo*' old # displays lines containing "f", or "fo", or "foo",
grep -	E '^fo*' old # shows lines starting with "f", or "fo",
# disp	lays lines ending with Svar:
grep -	E ''Svar'S' old
	E "SvarS" old

awk.	Script
script des	scribes the filtering actions by lines of the form:
	condition { instructions }
The awk utili	ity handles the input files one line at a time and executes instructions when the condition is true . If
condition is n	nissing, then instructions are run for all lines in the files.

awk. Instructions	
variable=expression	
* instructions if, for, while as in C	
• ; is instructions separator	
$^{\circ}$ the continuation of a line is done with the character $\mbox{\ensuremath{\backslash}}$ on the last position in the U	ine
• for (i in array) instruction <- is a repetitive structure in which i takes as values the array	ay's index
values and executes instruction for each value of i	
• through the expressions_list [>file_name] is displayed at the standard output (or in the	file
specified by file_name) the value of the expressions_list separated by OFS, and with O	RS at the
end of the line.	

awk.	Accessing fields
\$0	the whole line that is being processed
\$1	the first word/field on the current line being processed
\$2	the second
\$NF	The last word/field on the current line

