

Sisteme de Operare 1 - Curs 4

Curs tinut in 2012-2013 de catre lector dr. Sanda-Maria Dragos

Filtre Unix

orice comanda care citeste un fisier de la intrarea standard, il transforma si il afiseaza la iesirea standard;

sed

```
sed [-n] [-e scenariu ] [ lista_fisiere ]  
    [-f fisier_scenariu ]
```

(stream editor = considerat un editor de text neinteractiv)

prelucrează fişiere interpretate text conform unui scenariu sed. Prelucreează linie cu linie folosind un buffer temporar si afişează bufferul temporar la ieşirea standard (dacă nu se foloseşte opţiunea -n (, --quiet, --silent)).

Un scenariu sed este format din linii de forma

```
conditie instructiune
```

CONDȚII:

condiție vidă
adevarată pentru toate liniile din fişier;
n

slide 2



slide 3



adevarată pentru linia cu numărul de ordine egal cu n (liniile se numerotează cumulat în lista de fișiere)

\$

condiție adevarată pentru ultima linie din fisier

/expresie regulara/

condiție adevarată pentru liniile care conțin cel puțin un subșir care se potrivește cu expresia regulara

expr1,expr2

adevarată pentru liniile aflate între linia care se potrivește cu expr1 și linia care se potrivește cu expr2

```
sed 1,10 instructiune fis1
# execută instructiune asupra liniilor de la 1 la 10
sed 10,$ instructiune fis1 fis2
# execută instructiune asupra liniilor de la 10 la sfarsitul
5. # fisierului obtinut prin concatenarea lui fis1 cu fis2
```

slide 4



slide 5



INSTRUCȚIUNI:

p

afișează bufferul temporar la ieșirea standard

```
sed p fis1 # afișează fiecare linie de doua ori
sed -n p fis1 # afișează fiecare linie o data
```

d

șterge zona temporară

i\<ENTER>

are ca parametru un text (dat pe liniile următoare în fișierul scenariu) pe care îl afișează la ieșirea standard

```
fis-scenariu:
```

```
i\
```

```
ttttt\
```

```
11111
```

5.

```
sed -f fis-scenariu fis1
```

a\<ENTER>

analog cu \i dar afișează după prelucrarea fiecărie linii

y/sir1/sir2/

(unde sir1 și sir2 au lungimi egale)

realizează o translație înlocuind caracterele din fișierele de intrare care se găsesc în sir1 cu caracterele corespunzătoare din sir2

s/expresie_regulara/sir/[flaguri]

înlocuiește prima apariție a unui șir care se potrivește cu expresia regulată cu șirul "sir"

FLAGURI:

nimic

înlocuiește numai prima apariție

număr n între 1 și 512

înlocuiește exact a n-a apariție

g

înlocuiește toate aparițiile din linie

p

afișează buferul tampon la ieșire dacă s-a produs vreo modificare în linia respectivă

```
echo Sunday|sed s/day/night/  # Sunnight  
sed s/day/night old > new #sunbstituie prima aparitie la lui day cu night
```

```
# daca vreau sa inlocuiesc /usr/local/bin cu /common/bin
```

```
5. sed 's/\/usr\/local\/bin\/common\/bin/' old > new
```

```
sed 's_/usr/local/bin_/common/bin_' old > new
```

```
sed 's|/usr/local/bin|/common/bin|' old > new
```

```
sed 's:/usr/local/bin:/common/bin:' old > new
```

```
10. echo abcd123 | sed 's/\([a-z]*\).*\/\1/'
```

```
echo abcd123ef | sed 's/\([a-z]*\).*\/\1/' # afiseaza numai primul grup de litere
```

```
# interschimba primele 2 cuvinte
```

```
echo abcd efg | sed 's/\([a-z]*\) \([a-z]*\)/\2 \1/'
```

```
15.
```

```
# elimina spatiul dintre cuvinte
```

```
echo abcd efg | sed 's/\([a-z]*\) \([a-z]*\)/\1\2 /'
```

```
# spatiu alb = spatiu, TAB
```

```
20. sed 's/^[ \t]*//g' fis # elimina toate spatiile albe de la inceputul liniilor
```

```
sed 's/[ \t]*$/g' fis # elimina toate spatiile albe de la sfarsitul liniilor
```

```
# elimina toate spatiile albe de la inceputul si sfarsitul liniilor
```

```
sed 's/^[ \t]*//g;s/[ \t]*$/g' fis
```

```
25. sed /DA/s/bada/banu/g old # face inlocuirea numai in liniile care contin DA
```

```
echo "123def" | sed 'y/123456/ABCDEF/'
```

```
# sa afiseze numai liniile care s-au modificat
```

```
30. sed -n '/baz/s/foo/bar/p' old
```

grep

- caută un anumit șir de caractere într-un fișier sau în mai multe fișiere și afișează la ieșirea standard rezultatul. Denumirea vine de la expresia engleză „global/regular expression/print” care s-ar traduce prin: „tipărește expresie regulată globală”.

<pre>grep [-chilnsvw] [[-e] expresie_regulară -f nume-scenariu] [lista_fisiere]</pre>	slide 6
<pre>grep "ceva" fis grep -c "ceva" fis # 3 grep "ceva" fis f2 # la fiecare inceput de linie # se specifica numele fisierului din care linia face parte 5. grep -h "ceva" fis f2 # nu se mai scrie numele fisierului grep -l "ceva" fis f2 f3# fis</pre>	<div>?</div>
<div>slide 7</div> <div><div>?</div></div>	

Expresie regulară

Expresia regulară este o secvență de caractere în care unele caractere au semnificație specială. Cu caracterul "\" se evită semnificația specială a unui caracter, "..." evită orice caracter mai puțin \$ și '...' iar cu '...' se evită orice caracter.

<ul style="list-style-type: none">• <i>orice caracter</i>
[sir_caractere] <i>orice caracter din șirul de caractere</i>
[c1 - c2] <i>orice caracter cuprins între caracterele c1 și c2 în ordine lexicografică</i>
[^ sir_caractere] <i>negatia lui [sir_caractere]</i>
^ <i>dacă e primul caracter din expresia regulară semnifică început de linie</i>

```
grep ^if fis1 fis2 # afișează liniile din fis1 și fis2 care încep cu if
grep if fis1 fis2 # afișează toate liniile din fis1 și fis2 care
# conțin secvența "if"
```

\$

dacă e ultimul caracter din expresia regulată semnifică sfârșit de linie

```
grep fi$ fis1 fis2 # afișează liniile din fis1 și fis2 care se termină cu fi
```

\<

semnifică început de cuvânt (un cuvânt este format din litere, cifre sau -, orice alt caracter este considerat separator)

\>

semnifică sfârșit de cuvânt

*

repetă caracterul anterior interpretat ca expresie regulată de oricate ori

\{n\}

unde n este un număr între 0 și 255 repetă expresia anterioară de exact n ori

\{n,\}

repetă expresia regulată anterioară de cel puțin n ori

\{n,m\}

repetă expresia regulată anterioară de cel puțin n ori și de cel mult m ori

\(expr-regulată \)

<=> expr-regulată

\n

unde n este între 0 și 512

\n înlocuiește un șir cu care s-a înlocuit cea de-a n-a expresie regulată aflată între paranteze

```
grep '^\(.*\) \(.*\) \1$' fis1 # afișează toate liniile care încep și se
# termină cu același cuvânt separator fiind în spațiu
# și conțin mai mult de două cuvinte
```

```
grep 'o\{3\}' old # old: night is night
```

```
grep 'o\{2,\}' old # baz fooo bar
```

```
# foo bar baz
```

```
grep '\<fo*\>' old # foo bar
```

5.

```
grep '^<fo*\>' old
```

```
grep '\<'$var'\>$' old
```

```
grep "\<$var\>$" old
```

awk

Acest utilitar prelucrează fișiere text, selectand acele linii din text care satisfac condiții impuse de o listă de șabloane (expresii regulate) indicate la apelul utilitarului. Numele lui vine de la cei trei proiectanți și implementatori ai lui: A. Aho, P. Weinberger și B. Kerninghan.

Fisierul scenariu - descrie acțiunile de filtrare. Este descris prin linii de forma:

conditie { instructiuni }

utilitarul **awk** tratează pe rand ce o linie din fișierele de intrare și pentru fiecare execută instructiuni atunci cand conditie ia valoarea true. Dacă conditie lipsește atunci se execută instructiuni pentru toate liniile din fișiere.

Conditie

este o expresie logică construită cu operatorii din C: ||, &&, !, (). Operanzii pot fi expresii aritmetice, expresii relationale, constante și variabile. Variabilele nu trebuie (să fie) declarate, ele se inițializează automat, tipul lor deducadu-se din context. Pentru variabilele de tip șir de caractere există operatorul de concatenare (spatiu) precum și ceva funcții de lucru cu șiruri. Se pot folosi variabile de tip tablou ale căror indici pot să fie numerici sau șiruri de caractere.

Condiții predefinite

BEGIN
este adevarată inainte de prima linie din primul fișier

END
este adevarată după ultima linie din ultimul fișier

Instructiuni

- » variabilă=expresie
- » instrucțiunile if, for, while ca si in C
- » ; este separator de instrucțiuni
- » continuarea unei linii se face cu caracterul \ pe ultima poziție din linie
- » for (i in numetablou) instrucțiune



i ia ca valori indicii lui numetablou .i se execut. instrucțiune pentru fiecare valoare a lui i
» prin lista-expresii [>nume-fis]
afișează la ieșirea standard (sau in fisierul specificat prin nume-fis) valoarea expresiilor
separate prin OFS, iar la sfârșit de linie pune ORS.

Variabile predefinite

NF

numărul de cuvinte din linia curentă

NR

numărul de ordine al liniei curente (numărătoarea incepe de la 1); linia cu nr. 1 este prima linie din primul fișier

FNR

numărul de ordine al liniei curente; liniile cu nr. 1 sunt primele linii din fiecare fișier; numaratoare cepe de la 1 la ceputul fiecărui fișier

FS

separator de campuri

FILENAME

numele fișierului curent care este tratat

OFS

separator de campuri la ieșire (implicit este spațiu)

ORS

separator de inregistrări la ieșire (implicit este linie nouă)

ARGV

șirul parametrilor din linia de comandă

ARGC

numărul parametrilor din linia de comandă

Funcții predefinite

length(sir)

lungime sir; length <=> length(\$0)

substr(s,p,n)

subșirul lui s care cepe la poziția p și are lungimea n

index(s1,s2)

intoarce poziția la care s2 apare in s1 sau 0 la absență

sprintf(format, arg1,..)

intoarce ca rezultat șirul pe care printf l-ar tipări in C

split(s,a,c)

unde s este șir, a este tablou și c un caracter. Imparte șirul s în campuri considerand ca separator caracterul c dacă c lipsește atunci separatorul implicit este FS. Valorile împărțite sunt date ca valori elementelor tabloului a.

Accesarea campurilor se face cu \$1, \$2 ...\$i, \$(i+1), \$NF, iar intreaga linie se referă cu \$0

```
# afiseaza dintr-un fisier primul cuvant din fiecare linie
```

```
awk '{print $1}' fis
```

```
# afiseaza Salut anca de numarul de linii al fis
```

```
5. awk -v v=anca '{print "Salut ", v}' fis
```

```
# afiseaza toti utilizatorii din sistem care nu au parola
```

```
awk -F: '$2==" " { print $5}' /etc/passwd
```

```
10. # sa se afiseze numarul de linii pentru fiecare fisier prelucrat
```

```
{F[FILENAME]++}
```

```
END {for (f in F) print f, ":", F[f]}
```

```
# sa se afiseze numarul de caractere pentru fiecare fisier prelucrat
```

```
15. {F[FILENAME]+=length($0)}
```

```
END {for (f in F) print f, ":", F[f]}
```

```
# sa se afiseze dintr-un fisier liniile care concid
```

```
$0==v { if(NR>1) print $0 }
```

```
20. { v=$0 }
```


```
# sa se afiseza toate cuvintele dintr-un fisier si numarul lor de aparitie
```

```
{for(i=1; i<=NF; i++) X[$i]++}
```

```
END {for (c in X) print "cuvantul", c, "apare de ", X[c], "ori!" }
```


sort

sorteaza lexicografic liniile unui fisier text

<pre># listeaza continutul directorului # curent ordonat lexicografic dupa grup ls -l sort -k8</pre>	slide 9
<pre>5. # listeaza ... ordonat numeric si # descrescator dupa dimensiune ls -l sort -rnk5</pre>	

--	--

tr	slide 10
(translate) - traduce sau sterge caractere	



<pre>echo "san@she^jduj~1" tr "@^~" "_=/"</pre>	
<pre># -s reduce la un singur caracter aparitia # repetata a unui caracter din sir2</pre>	
<pre>5. echo "san@she^jduj~1" tr -sd "@^~" " "</pre>	
<pre># -d elimina la iesire caracterele care # apar in sir1</pre>	
<pre>echo "san@she^jduj~1" tr -d "@^~"</pre>	

wc	
(word count) - numara caracterele, liniile sau cuvintele din fiecare fisier	

<pre>echo "sanda@shdfgshe^jduj~1" wc -c</pre>

head, tail
<i>head</i> - reda primele n linii dintr-un fisier
<i>tail</i> - reda ultimele n linii dintr-un fisier

<pre>head -10 /etc/passwd</pre>
<pre>tail -20 /etc/passwd</pre>

Comenzi de gestiune exterioara a proceselor	slide 11
tee	
ajuta la obtinerea unui fisier martor al iesirii standard pentru o anumita comanda	
comanda tee [-ia] fisier	
<div><div><div><div><div><div>-i</div><div>ignorarea intreruperilor pe timpul derularii comenzii</div></div><div><div>-a</div><div>iesirea comenzii sa fie adaugata la fisier</div></div></div></div></div></div>	
nice	slide 12
Exista 19 ordine de prioritate in servirea proceselor din sistemele Unix, numerotate de la 1 la 19. Prioritatea minima este 19, iar cea maxima este 1. Daca nu se fac nici un fel de precizari privind prioritatile, atunci toate procesele vor fi rulate cu prioritatea 10, sau cu o prioritate implicita fixata de printr-un apel sistem.	
nice [-n] comanda	
indica sistemului de operare sa execute comanda cu o prioritate mai slaba decat cea implicita. + {n : 10}	
<div><div># executa sortarea fisierului A cu prioritatea 18</div><div>nice -8 sort <A >B &</div></div> <div><div># Superuserul are posibilitatea de a lansa nice cu o valoare negativa:</div><div>5. nice --10 rm -r /*.TMP <A >B & # comanda se va executa cu prioritate maxima</div></div>	

nohup

rularea comenzii cu imunitate la deconectare (adica la CTRL-d); dupa delogare comanda poate continua sa ruleze in background.

```
$nohup comanda
```

kill

emite un semnal de tip intrerupere catre un proces.

```
kill [-semnal] PID
```

semnal- un numar intre 1 si 32 (9 - oprirea neconditionata; 15 - semnal software de oprire - IMPLICITA)

```
kill -9 PID
```

ps

afiseaza starile unui proces

```
ps [-al] [-t terminale]
```

- a afisearea starilor pentru toate procesele active din sistem
- l formatul lung de listare
- t numai procesele lansate de la anumite terminale

slide 13



slide 14



```
# PID, TTY, TIME(servit de CPU), CMD
ps -a | less # afiseaza toate procesele din sistem


5. ps -al | less # afiseaza ... in format lung ()

ps -U sanda -u sanda u # afiseaza toate procesele
# utilizatorului (real si efectiv) sanda

ps -U sanda -u sanda a

10. ps -U root -u root -N # afiseaza toate procesele care nu sunt ale utiliz. (real s

ps -lt pts/1, pts/2
```

Aplicatii Shell	slide 15
sup - aplicatie de supraveghere a modificarii continutului unui director;	
<pre>#!/bin/sh DIR=\${2-\${HOME}} t=\${1-60} 5. x=`ls -l \$DIR` while true do 10. sleep \$t y=`ls -l \$DIR` if ["\$x" != "\$y"] then echo "Directorul \$DIR modificat!" 15. exit fi x=\$y done</pre>	
<pre>\$ chmod 755 sup \$./sup 10 ..</pre>	