.::shell scripts::.

d u n e t n a . k e r n e l p a n i c

Índex

0. sarrera

Manual hau Kernelpanic-en (Bartzelonako hacklab bat) “Shell script-en programazioa”-ri buruz egindako kurtso batetan oinarrituta dago

Bere helburua ez da izugarrizko manual bat izatea, soilik laguntza bezala erabiltzea pertsona haiek, zeinek programatzen jakinda eta UNIX buruz ezagupen batzuk izanda, erabil nahi dutenek shell lengoaia script-en bidez bizitza errazteko

Lan hau CSO Les Naus-i dedikatzen dugu, 2003-ko abenduak 9 desalojauta izan zelarik. Txokoak kenduko dizkigute, baina borrokan jarraituko dugu, eraikitzen eta ezagupen eta ilusioak banatzen ere.

NOTA: Parentesisen artean dauden script-en adibideak http://kernelpanic.hacklabs.org orrialdean topatu daitezke. Edozein zalantza edo gomendiorako posta elektroniko bat bidali dezakezue hurrengo helbidera: info@kernelpanic.hacklabs.org.

1. oinarrizko kontzeptuak

1.1. aldagaiak

Datuak gordetzeko shell-aren bidez aldagaiak definitu ditzakegu. Memoriako bi zona ditugu non aldagaiak definitu ditzakegu: ingurune eta lokalen barrutia.

Aldagai lokalak: lan egiten ari garen shell-agatik ikusi daitezke soilik, ikusiezina dira beste subshell-entzat, hau da, shell-aren edozein prozesurako ikusezinak dira.

Inguruko aldagaia: lan egiten ari garen shell-ak zein edozein subshell batentzat ikusgarriak dira (edo shell-eko edozein subprotzesurako).

NOTA: haur protzesu batean deklaratutako aldagai bat ikusezina izango da bere aita protzesurako (nahiz eta inguruko aldagaia izanda)

KOMANDUAK	
set	Definitutako aldagai guztiak ikusteko
env	Definitutako inguruko aldagai guztiak ikusteko
aldagai_izena=balioa_var	Aldagai bat definitu eta balio bat erantsi
export *aldagai_izena=balioa_var*	Inguruko aldagai bat definitu eta balio bat erantsi
export *aldagai_izena*	Aldagai lokal bat inguruko aldagai batean bihurtu
unset *aldagai_izena*	Aldagai bat askatu

::adibidea::

```
$ nerealdagaia=3          aldagai lokal bat deklaratzen dut
$ echo $nerealdagaia      bere balioa begiratzen dut
3
$ bash                    subshell batean sartzen naiz
$ echo $nerealdagaia      lokala denez, ez dauka baliorik
$ exit                    shell nagusira itzultzen gara
$ export nerealdagaia     aldagai lokala inguruzko aldagaian
                          aldatzen dut
$ echo $nerealdagaia      shell nagusian bere balioa ikusten
3                         jarraitzen dut
$ bash                    subshell batean sartzen naiz
$ echo $nerealdagaia      orain bai ikusten dugu bere balioa!!!
                          (ingurukoa bait da)
$ exit
```

Aurre definitutako aldagaiak: Aurrez definitutako zenbait aldagai erabil ditzakegu edozein motako informazioa jaso edo uzteko.

AURRE-DEFINITUTAKO ALDAGAIAK	
HOME	Erabiltzen ari garen lan direktorioa
PATH	Path-a idatzi gabe sartu gaitezken lekuak
PS1	Prompt primarioa
PS2	Bigarrenezko prompt-a
BASH	Bash programaren path-a
BASH_VERSION	Erabiltzen ari garen bash-aren bertsioa
COLUMNS	Pantailan agertzen diren zutabe kopurua
GROUPS	Erabiltzailearen talde nagusiaren idenfikadorea
HISTCMD	Komandu honen istorikoaren indexa
HISTFILE	Istorikoa gordetzen den fitxategia
HISTFILESIZE	Istoriko fitxategiaren luzaera
HOSTNAME	Makinaren izena
LANG	Edozein LC_-n ez bada esaten, lehenetsitako hizkuntza
LC_ALL	Hizkuntza
PID	Momentu honetan aritzen den protzesuaren identifikadorea
PWD	Oraintxe gauden path-a
RANDOM	Zenbaki aleatorioa
SECONDS	Makina pizturik daraman segunduak
UID	Unezko erabiltzailearen identifikadorea

::adibidea::

$ echo “Nere lan direktoria $HOME da eta nere talde id-a $GROUPS da”
Nere lan direktoria /home/kp da eta nere talde id-a 1000 da

1.2. aldagaien ordezketa

Aldagaien aldaketa: Aldagai baten edukiaren balioari erreferentzia egiteko erabiliko dugun teknika da.

ALDAGAIEN ORDEZKETA	
$*aldagai_izena*	*aldagai_izena* aldagaiaren balioari erreferentzia egiten du
${*aldagai_izena*}	*aldagai_izena* aldagaiaren balioari erreferentzia egiten du. Aldagaiaren izena giltzen tartekoa mugatzen gaitu.

::adibideak::

1::: $ agurtu=”kaixo” “kaixo” balioarekin definitzen dut *agurtu* aldagaia
$ izena=”anna” “anna” balioarekin definitzen dut *izena* aldagaia
$ echo agurtu izena — aldagaien balioak pantalairatzen ditut
agurtu izena — gaizki dago: aldagaien izenak jarri ditut eta ez bere balioa.
$ echo $agurtu $izena — orain ondo dago!
kaixo

2::: $ masc=”gat” “gat” balioarekin definitzen dut *masc* aldagaia
$ echo masculí:$masc femení:$masca $masc pantailaratzen dut eta $masc *a* batengatik segiturik

```
                                    erakusten du.
   masculí:gat femení:              masca, aldagai (huts) bat
                                    bezala hartzen du
$ masculí:$masc femení:${masc}a     {} jartzen dut aldagaiaren
                                    izena desberdintzeko
masculí:gat femení:gata             horixe da ni nahi nuena!!!
```

1.3. komandoen aldaketa

Komandoen aldaketa: Komandu baten irteera komandu baten izenagatik aldatzeko erabiltzen dugun teknika da.

KOMANDOEN ALDAKETA	
`*komandu*` **$(*komandu*)**	Komandoaren izena bere irteerakin ordezkatzen du

::adibideak::

1:::
```
# whoami            komandu bat exekutatzen dugu
root                zelan bezala konektauta zauden esaten diguna
# echo Ni whoami naiz       pantalairatu nahi badugu eta komando
                            hau jarri ezkero...
Ni whoami naiz              ...jarri duguna literalki aterako da
# echo Ni `whoami` naiz     aldatzen dugu whoami bueltatzen duen
                            balorearikin eta...
Ni root naiz                ...nahi genuena lortzen dugu!!!
# echo Ni $(whoami) naiz    beste modu batez
Ni root naiz
```

2:::
```
$ date +%F; date +%D        %F eta %D formatuekin exekutatzen dugu
2004-10-21                  date
10/21/04
$ aaaammdd="%F"             formatu hauen balioa bi aldagaiei
$ mmddaa="%D"               ematen dizkiegu
$ echo "Data (aaaa-mm-dd):`date +$aaaammdd`"  komanduak eta
                                    aldagaiak aldatzen ditugu...
Data (aaaa-mm-dd):2004-10-21
$ echo "Data (mm/dd/aa):`date +$mmddaa`"
Data (mm/dd/aa):10/21/04
```

3::: gure lan direktorioaren segurtasun kopia bat egiten duen script txiki bat:

```
CS=/var/backup-`date +%Y%m%d`.tgz  segurtasun kopiaren
                                   izenarekin izendatutako
                                   aldagaia
                                   (ex: /var/backup-20040601)

tar -czf $CS /home/nomusu  lan direktorioa konprimatu eta trinkotu
```

1.4. karaktere bereziak

Karaktere bereziak: shell-entzat karaktere batzuk esanahi berezia dute. Zenbait teknika daude shell-ak kontuan izateko edo kasurik ez egiteko.

KOMANDUAK	
\	atzean doan karakterearen esanahi berezi deuseztatzen du
' '	komila artean dauden karaktere guztien esanahi berezia deuseztatzen du
" "	karaktere guztien esanahi berezia deuseztatzen du, $ \ `` ""kenduta

::adibideak::

```
1::: $ echo "El "silenci""        "" artean silenci pantailaratu nahi
                                  dugu...
     El silenci                   karaktere berezi bezala hartzen ditu "
     $ echo El \"silenci\"        "escapatu" behar ditugu
     El "silenci"

2::: # echo 'Ni $LOGNAME naiz eta $PWD-n nago'   ''-ekin...
     Ni $LOGNAME naiz eta $PWD-n nago            $ ez du hartzen
     # echo "Ni $LOGNAME naiz eta $PWD-n nago"  ""-ekin...
     Sóc el/la root i estic a /root              ...bai hartzen ditu!
     # echo "Ni $LOGNAME naiz eta \$PWD: $PWD"  barrejant "" i \ ...
                                                 "" eta \ nahasten...

     Ni root naiz eta $PWD: /root        ...komeni bazaigu,
                                         har ditzazkegu ala ez
```

1.5. s/i-en berbideratzea

***stdin*, *stdout* i *stderr*:** Zenbait komandu estandar sarreratik onartzen dituzte datuak (stdin, 0 fitxategiaren deskriptorearekin), hau da, teklatutik. Baita ere, zenbait komandu irteera estandarretik ematen dizkigute (stdout, 1 fitxategiaren deskriptorearekin), hau da, pantaila. Azkenez, komandu batek egin ditzakeen errore guztiak errore irteerara bideratzen dira (stderr, 2 fitxategiaren deskriptorearekin).

s/i-en berbideratzea: s/i-ren berbideratzeak hartutako sarrera, irteera ala erroreen datuak lehenetsitako fitxategi (*stdin*, *stdout*, *stderr*) desberdin batean berbideratzea uzten digu

S/I-EN BERBIDERATZEA	
<	*stdin-en* berbideratzea
>	*stdout-en* berbideratzea fitxategia existitzen ez bada, sortzen du fitxategia existitu ezkero, edukia kargatzen da
>>	*stdout-en* berbideratzea fitxategia existitzen ez bada, sortzen du fitxategia existitu ezkero, ondoren eransten du
2>	*stderr-en* berbideratzea fitxategia existitzen ez bada, sortzen du fitxategia existitu ezkero, edukia kargatzen da
2>>	*stderr-en* berbideratzea fitxategia existitzen ez bada, sortzen du fitxategia existitu ezkero, ondoren eransten du
1>&2	*stdout stderr*-era berbideratu
2>&1	*stderr stdout*-era berbideratu
>&	*stdout* eta *stderr* fitxategi batera berbideratu

::adibideak::

1::: $ mail root -s "hack your mind!" < mailpr *mailpr* mezua duen fitxategia izango zen

2::: $ cat fit1 fit2 fit1 existitzen da fit2 ez da existitzen

hau *fit1*-aren edukia da *stdout*
cat: fit2: No existe el fichero o el directorio *stderr*

3::: $ cat fit1 fit2 >> copiafit 2>> error.log
fit1 copiafit fitxategian eransten dugu eta errorea *error.log*-ean gordeko da

4::: $ cat fit1 fit2 2> fit3 1>&2 (*fit1* existitzen da, *fit2* ez da existitzen,
irteera guztiak *fit3*-n daude, zeren erroreak *fit3*-n gordetzen dira eta *stdout stderr*-era berbideratzen ditugu

$ cat fit1 fit2 > fit3 2>&1 (*fit1* existitzen da, *fit2* ez da existitzen,
irteera guztiak *fit3*-n daude, zeren stdout *fit3*-ra badoa eta *stderr stdout*-era berbideratzen ditugu

5::: $ cat fit1 fit2 >& fit3 (*fit1* exititzen da, *fit2* ez)
irteera guztiak *fit3*-n daude, zeren stdout eta stderr *fit3*-ra berbideratzen ditugu

1.6. iragazkiak

Iragazkiak: stdin emandako datuak hartu eta stdout-tik datuak ateratzen dituen programa bat da, stdin-etik sartutako datuak aldatu gabe.

IRAGAZKIAK

cat [-n] [*fitxategiaren_izena*]

stdin/fitxategiaren_izena erakusten du
-n lerroak zenbakitzen ditu

head [*-zen*] [*fitxategiaren_izena*]

fitxategi/stdin-en aurreneko *zen* lerroak erakusten ditu (lehenetsita 10)

tail [*-zen*] [*fitxategiaren_izena*]

fitxategi/stdin-en azkeneko *zen* lerroak erakusten ditu (lehenetsita 10)
-f fitxagia aldatzen den heinean irteera dinamikoa da

wc [-lwc] [fitxategiaren_izena]

fitxategiaren_izena/stdin-en lerro, hitz eta karaktereak kontatzen ditu
-l lerro kopurua soilik
-w hitz kopurua soilik
-c karaktere kopurua soilik

cut -c*lista* [*fitxategiaren_izena*]

lista-n aipatutako zutabeak ateratzen ditu
lista-ren formatua:
A,B A eta B zutabe/eremuak aukeratu
A-B *A-tik B-raino zutabe/eremuak aukeratu*
A- A zutabe/eremutik bukaeraraino
-B *hasieratik B zutabe/eremuraino*

cut -f*lista* -d*sep* [*fitxategiaren_izena*]

lista-n aipatutako eremuak atera *sep* bereizlearen arabera
lista-ren formatua aurrekoaren berdina da

grep [-cinv] *maisua* [*fitxategiaren_izena*]

maisu baten arabera fitxategi/stdin lerroen bilaketa
-c lerroaren zenbakia bakarrik erakusten du
-i maiuskula/minuskula ez du kontutan hartu
-n lerroaren zenbakia eransten du
-v maisua jarraitzen ez duten lerroak erakusten ditu
Maisuaren formatua (oinarrizko adierazpen erregularrak):
. edozein karaktere sinplea
[] karaktere multzo bat
[^] kortxete artean ez dagoen edozein karaktere
[-] barrutiak
* 0 edo agerpen gehiago aurreko adierazpenean
+ 1 edo agerpen gehiago aurreko adierazpenean
^*exp* *exp*-ez hasten den edozein kate
exp$ *exp*-ez bukatzen den edozein kate

tr *c1 c2* [*fitxategiaren_izena*]

fitxategiaren_izena/stdin-en *c1 c2*-gatik itzultzen du

tr -s *c1* [*fitxategiaren_izena*]

ondoz ondoko c1-ak bat bakarrean bihurtzen du

sed 's/*expr1*/*expr2*/[g]'

expr1 expr2-gatik ordezkatzen du
g agerpen guztiak ordezkatzen ditu

sed -r 's/*expr1*/*expr2*/[g]'

expr1 expr2-gatik ordezkatzen du adierazpen erregular konplexukin

IRAGAZKIAK
g agerpen guztiak ordezkatzen ditu
sort [-ndutsep] [-k *num*] [*fitxategiaren_izena*]
fitxategiaren_izena/stdin-ren lerroak zerrendatzen ditu -n zenbakizko zerrendatzea -d letra, zenbaki edo hutsak ez badira ez ditu kontuan hartzen -u bikoiztutako lerroak ez ditu kontuan hartzen -tsep eremu mugatzaile bat zehazten du -knum *num* eremuagatik zerrendatuko dugula zehazten du
uniq [*fitxategiaren_izena*]
fitxategiaren_izena/stdin-en zenbait lerro berdin eta ondoz ondokoak bat bakarrean bihurtzen du
tee [-a] *fitxategiaren_izena1* [*fitxategiaren_izena2*]
fitxategiaren_izena2/stdin pantailaratzen du eta *fitxategiaren_izena1*-en idazten du *-a fitxategiaren_izena1*-en gainidatzi ordez ondoren eransten du

::adibideak::

1::: /var/log/messages fitxategitik erakutsi:

```
    a:: fitxategi osoaren edukia
        $ cat /var/log/messages
    b:: lehendabiziko 3 lerroak
        $ head -3 /var/log/messages
    c:: azkeneko lerroak aldatzen diren heinean
        $ tail -f /var/log/messages
    d:: lerro kopurua
        $ wc -l /var/log/messages
```

2::: /etc/passwd fitxategitik erakutsi:

```
    a:: sistemaren erabiltzaileen lehendabiziko 3 letrak
        $ cut -c1-3 /etc/passwd
    b:: erabiltzaileen izena eta lan direktorioa
        $ cut -d: -f1,6 /etc/passwd
```

3::: Hurrengoa betetzen duten erabiltzaileak erakutsi:

```
    a:: m edo M-gatik hasitako erabiltzaileak
        $ grep -i ^m /etc/passwd
    b:: 1000 eta 1999 artean id-a dutenak
        $ grep ^.*:x:1...: /etc/passwd
    c:: shell bezala /bin/bash dutenak
        $ grep :/bin/bash$ /etc/passwd
    d:: beste edozein shell-a dutenak
        $ grep -v :/bin/bash$ /etc/passwd
```

4::: /etc/passwd fitxategiaren edukia pantailaratu:

```
    a:: eremuak – batekin bereizirik
        $ tr ":" "-" /etc/passwd
    b:: koma guztiak bakarrean bihurtu
        $ tr -s "," /etc/passwd
    c:: /bin/bash shell-a /bin/sh-gatik aldatu
        sed 's/\/bin\/bash/\/bin\/sh/' /etc/passwd
    d:: erabiltzaile izenaren arabera zerrendatu
        sort /etc/passwd
```

e:: erabiltzailearen identifikadoreagatik zerrendatu
sort -nt: -k3 /etc/passwd

1.7. hodiak (pipelines)

Hodia: komandu baten irteera, beste komandu baten sarrera berbideratzeko balio du.

KOMANDUAK	
komandu1* | *komandu2	*komandu1*-aren *stdout* *komandu2*-ren *stdin*-era berbideratzen dugu

::adibideak::

1::: Sistemaren erabiltzaileen izenak eta bere lehenetsitako shell-a erakutsi.
$ cat /etc/passwd | cut -f1,7 -d: | sort | tr ":" "\t"

2::: /etc/profile eta /etc/hosts fitxategien lerroen batuketaren emaitza pantailaratu
$ wc -l /etc/profile /etc/hosts | tail -1 | cut -f2 -d" "

3::: Direktorio honetan urtarrilan aldatutako fitxategi kopurua (direktorioak kenduta), eta aldi berean "urtarrila" izendatutako fitxategi batean fitxategi hoien izena idaztea
$ ls -l | grep -v ^d | tr -s " " | cut -f6,8 -d" " |
grep ^....-01- | cut -f2 -d" " |tee urtarrila | wc -l

4::: /etc/passwd fitxategia hartuz gero, *neretaldea* izendatutako fitxategi bat sortu zeinek zure talde berdineko erabiltzailearen izena, hastapeneko shell-a eta hastapeneko direktorioak idazten duena.
$ cat /etc/passwd | grep ^.*:.*:.*:`id -g`: | cut -f1,7,6 -d: > neretaldea

5::: $ ps -ef | grep "^root " | tr -s " " | cut -f2,8 -d" " > procsroot
procsroot izendatutako fitxategi bat lortu zeinek root erabiltzailearen prozesu guztien identifikadore (PID) eta izena (CMD) gordetzen dituena.

6::: *propsetc* izendatutako fitxategi bat lortu zeinek /etc direktorioan dauden fitxategi eta direktorioen jabeen lista bat (errepikatu barik) duena
$ ls -l /etc | tr -s " " | cut -f3,4 -d" " | sort | uniq > propsetc

2. shell script-en programazioa

2.1. shell script-en exekuzioa

Shell script: shell-aren lengoai propioa erabiltzen duen programa bat besterik ez da. Exekutatzeko modu desberdin daude:

1. shell aktiboaren subshell batean
   a) $ bash *script-aren_izena*
   b) $ *./script-aren_izena* (exekuzio baimenak behar ditugu)
   c) $ *script-aren_izena* (exekuzio baimenak behar ditugu eta *script-aren_izena* dagoen direktorioa PATH-ean egotea)

2. shell-aren seme prozesu bezala exekutatzen den programa_exekutable bat bezala
   scriptaren aurreneko lerroan idatziko dugu zein shell-akin interpretatu nahi dugu:
   ```
   #!/bin/bash
   ```
   eta script-a 1 kasua bezala exekutatzen dugu

3. shell aktibo berean
   $ source *script-aren_izena*
   $. script-aren_izena

::adibidea::

lo.sh izeneko script bat sortu hurrengo lerroarekin

```
sleep 222
```

1 motako exekuzioa:

```
$ chmod u+x lo.sh
$ ./lo.sh
beste kontsola batean ikusiko genuke:
$ pstree -p
   ...
   -bash(905)---bash(1542)---sleep(1543)
   ...
```

2 motako exekuzioa:

```
script-ak hurrengo edukia izango du:
#!/bin/bash
sleep 222
$ ./lo.sh
beste kontsola batean ikusiko genuke:
$ pstree -p
   ...
   -bash(905)---lo.sh(1564)---sleep(1565)
   ...
```

3 motako exekuzioa:

```
$ . lo.sh
beste kontsola batean ikusiko genuke:
```

```
$ pstree -p
   ...
   -bash(905)---sleep(1576)
   ...
```

2.2. shell script-en arazketa

Arazketa: Programa bat idazterakoan eta lortutako emaitzak ez badira itxaroten zirenak bezala, hori araztea uzten digun tresna bat izatea oso egokia da

ARAZKETA	
set -x	arazketa aktibatzen du
set +x	arazketa desaktibatzen du
set -e	agindu sinple bat exitu gabe bukatuz gero lehenbailen atera (itzul kode bat zero desberdina dena)
set +e	komandu bat exitu gabe irtetzen bat irteera desaktibatzen du

::adibidea::

Hurrengo script-a badugu:

```
AGURTU_K=”kaixo”
AGURTU_A=”agur”
set -x
echo “$AGURTU_K $LOGNAME”
echo “Konektaturik dauden erabiltzaileak”
who
set +x
echo “$AGURTU_A $LOGNAME”
```

“set -x” eta “set +x”-en artean dauden kode lerroen arazketa egingo du exekutatzerakoan eta hurrengo emaitza jasoko dugu pantailatik:

```
++ echo kaixo hm
kaixo hm
++ echo Konektaturik dauden erabiltzaileak
Konektaturik dauden erabiltzaileak
++ who
hm   tty1         Jun  3 19:39
++ set +x
agur hm
```

2.3. iruzkinak

Iruzkinak: Programatzerakoan kontutan izan behar dugu ez gaudela bakarrik munduan, edo nahiz eta egon, memoria failatu dezake eta ez ulertzea guk geuok aurrez idatzitako programa bat

IRUZKINAK	
#iruzkina	kodearen zatiak iruzkindu

2.4. parametroak eta aldagai bereziak

Parametroak: script bat exekutatzerakoan pasatzen dizkiogun baloreak dira eta programa barruan ikustea nahi dugu

$ ./*script-aren_izena parametro1 parametro2 ... parametron*

Hurrengo taularen bidez pasatutako parametroak erreferentziatu ditzakegu:

PARAMETROAK	
$0	exekutatzen ari den *shell-script*-aren izena
$*n*	shell-scripta-ari n posizioan pasatutako parametroa
"$*"	"par1 par2 ..." kate batean parametroak zabaltzen ditu
"$@"	"par1" "par2 ... kate desberdinetan parametroak zabaltzen ditu
shift *n*	n posiziotan parametroak mugitu. n parametro gabe posizio bat mugitzen du.

Aldagai bereziak: Gure beharren arabera edozein script-ean erabil ditzazkegun aurre-definitutako aldagaiak ditugu

ALDAGAI BEREZIAK	
$#	parametro kopurua
$$	exekutatzen ari den prozesuaren shell-aren PID-a
$!	exekutatutako azken prozesuaren PID-a
$?	exekutatutako azken prozesuaren itzultzeko kodea

::adibidea::

(sistema.sh) sistemaren datu desberdinez informatzen gaituen script-a

2.5. s/i instrukzioak

Sarrera eta irteeren instrukzioak: stdin (sarrera) bitartez irakurtzeko eta stdout-etik (irteera) erakusteko ahalmena duten instrukzioak dira.

KOMANDUAK	
read *aldagai_izena*	*stdin*-en bitartez sarrera
echo [-ne] *katea/$aldagai_izena*	*stdout*-en bitartez irteera bukaerako lerro saltuarekin -e \ duten karaktereak interpretatu \n : lerro saltoa \t : tabuladorea ... -n bukaerako lerro saltua ezabatzen du

::adibidea::

```
$ read nerealdagaia                    teklatuz lortutako aldagai bat
                                       irakurzen dugu
3
$ echo -e "Nere aldagaia:\t\t$nerealdagaia"     bere balioa
Nere aldagaia:          3                       formatu batekin
                                                ikusten dut
```

2.6. operadoreak

Zenbakizko operadoreek: zenbakiak duten aldagaiak eta zenbakiekin operatzen onartzen dutenak.

Operadore logikoak: baldintza konposatuak zehazteko balio dute

ZENBAKIZKO OPERADOREAK		OPERADORE LOGIKOAK	
+	batuketa	!	ez
-	kenketa	&&	eta
*	biderketa	||	edo
\/	zatiketa		
%	hondarra		
\(\)	parentesisa		

2.7. zenbakizko adierazpenen azterketa

Edonoiz zenbakizko kalkuluak egin behar ditugu, bai emaitza erakusteko bai aldagai batean gordetzeko.

ZENBAKIZKO ADIERAZPENAK	
zenb_adier	stdout-en bidez zenb_adier aztertu (hutsune batekin operandu operadoretik baztertu behar dugu zenb_adier-en)
let *zenb_adier* **((*zenb_adier*))**	zenb_adier aztertzen dugu (zenb_adier-en operandu operadoretik ez dugu hutsune batekin baztertu behar)

::adibideak::

```
1::: $ expr 3 + 5          3+5 kalkulatzen dugu
     8                     8!

2::: $ ((a=3+5))           3+5 kalkulatzen dugu eta a aldagaian gordetzen
                           dugu emaitza
     $ echo $a             aldagaiaren balioa erakusten dugu
```

```
     8

3::: $ a=1
     $ ((a=$a+1))        a-ren balioa batean gehitzen dugu
     $ echo $a           aldagaiaren balioa erakusten dugu
     2
```

2.8. baldintzen zehazpena

Baldintza: Programa baten fluxua apurtzeko baldintza batzuk zehaztu behar ditugu alde batera edo bestera bideratzeko. Balditzak zehazteko erabiltzen dena:

test *expr*
baita ere
[expr]

Ondoren, espezifikatu ditzazkegun baldintzak azaltzen dira:

FITXATEGIEN BALDINTZAK	
[-e *fit*]	*true* fitxategia existitzen bada
[-d *fit*]	*true* fitxategia existitu eta direktorio bat bada
[-f *fit*]	*true* fitxategia existitu eta erregularra bada
[-L *fit*]	*true* fitxategia existitu eta lotura sinbolikoa bada
[-r *fit*]	*true* fitxategia existitu eta irakurtzeko permisua badu
[-w *fit*]	*true* fitxategia existitu eta idazteko permisua badu
[-x *fit*]	*true* fitxategia existitu eta exekutatzeko permisua badu
[*fit1* -nt *fit2*]	*true fit1 fit2* baino berriagoa bada
[*fit1* -ot *fit2*]	*true fit1 fit2* baino zaharragoa bada
KATEEN BALDINTZAK	
[*kate*]	*true* katea hutsik ez badago
[-n *kate*]	*true* katearen luzera 0-ren ezberdina bada
[-z *kate*]	*true* katearen luzera 0 bada
[*kate1* = *kate2*]	*true kate1* eta *kate2* berdinak badira
[*kate1* != *kate2*]	*true kate1* eta *kate2* ezberdinak badira
ZENBAKI OSOEN BALDINTZAK	
[*zen1* -eq *zen2*]	*true zen1* eta *zen2* berdinak badira
[*zen1* -ne *zen2*]	*true zen1* eta *zen2 ez*berdinak badira
[*zen1* -gt *zen2*]	*true zen1 zen2* baino handiago bada
[*zen1* -ge *zen2*]	*true zen1 zen2* baino handiago edo berdin bada
[*zen1* -lt *zen2*]	*true zen1 zen2* baino txikiago bada
[*zen1* -le *zen2*]	*true zen1 zen2* baino txikiago edo berdin bada

2.9. egitura alternatiboak

Egitura alternatiboa: Baldintz bat bete ezkero, kode zati bat exekutatu ahal izatea baimentzen gaituenak dira

if EGITURA	**case EGITURA**

if *baldintza1*
then
  instrukzioak
elif baldintza2
then
  instrukzioak
else
  instrukzioak
fi
	case $*aldagai_izena* in
	  maisu1) *instrukzioak*;;
	  maisu*2*) *instrukzioak*;;
	  ...
	  maisu*n*) *instrukzioak*;;
	  *) *instrukzioak*;;
	esac

::adibideak::

1::: .bashrc fitxategiaren existentzia konprobatzen duen script-a (existrc.sh).

2::: Parametro baten bidez pasatutako fitxategia existitzen bada konprobatzen duen script-a (propsfit.sh)

3::: Parametro baten bidez pasatutako direktorioaren segurtasun kopia bat egiten duen script-a (backup.sh) eta /var/backups gordetzen du backup-aaaammdd.tgz formatua duen izena.

4::: Sistemara sartzerakoan eta ateratzerakoan soinu bat jotzeko /etc/init.d-en sartzeko pentsatutako script-a (soinifi.sh)

2.10. egitura errepikakorrak

Egitura errepikakorrak: kode zati bat zenbait bider errepikatzen uzten gaituenak dira.

while EGITURA	**until EGITURA**	**for EGITURA**

while *baldintza*
do
  instrukzioak
done
	until *baldintza*
	do
	  instrukzioak
	done
		for *aldagai_izena* in *balioen_lista*
		do
		  instrukzioak
		done

::adibideak::

1::: Uneko direktorioan dauden zenbait fitxategi berizendatzen dituen script-a (rename.sh).

2::: fitxategieen izenak argumentu bezala onartu eta bere edukia hasieran fitxategiaren izenaren goiburua erakusten duen script-a (llistafit.sh)

3::: parametro bezala pasatutako direktorioaren direktorio eta fitxategi zuhaitzaren ikerketa oso bat egiten duen script-a (estatdperm.sh) eta hurrengo informazioa lortzen dugu:

Irakurtzeko baimena dugun fitxatxegi kopurua

Idazteko baimena dugun fitxatxegi kopurua
Exekutatzeko baimena dugun fitxatxegi kopurua
Irakurtzeko baimena ez dugun fitxatxegi kopurua
Idazteko baimena ez dugun fitxatxegi kopurua
Exekutatzeko baimena dugun fitxatxegi kopurua

4::: 1 eta 75 artean dauden zenbakiak argumentu bezala jasotzen dituen script-a (grafika.sh). Argumentu bat barrutiaren barne ez badago errore bat emango du eta ezer egin gabe bukatuko du.
Barne badago, argumentu bakoitzeko argumentu balioaren arabera asteriskodun lerro bat sortuko du

5::: argumentu bezala pasatutako komandua PATH aldagaian dituen direktoriotan bilatzen duen script-a (nondago.sh), bere kokalekua adieraziz eta komanduaren deskripzio motz bat.

6::: parametro zela pasatutako hitz guztiak fitxategi batean idazten duen script-a (hitzfit.sh). hitza0, hitza1,... bezala izendatuko dira fitxategi hauek

7::: zu ordez kontatzen duen script-a (segida)

8::: parametro bezala pasatutako fitxategeei hasieran #!/bin/bash lerroa eransten duen script-a (binbash.sh), oraindik ez badu (memoria gutxidunentzat).

3. pixka bat gehiago...

3.1. taulak

Taula: Konposatutako datuen egitura bat da

bal0	bal1	bal2	bal3	bal4	bal5	bal6	bal7	bal8	bal9
0	1	2	3	4	5	6	7	8	9

arr

Ondorengo taulan zelan definitu eta nola erreferenziatzen den ikus dezakegu:

TAULAK	
arr_iz=(bal1 bal2 ... baln)	hasierako erazagupen eta esleipena
declare -a *arr_iz*	Taula baten erazagupena. Esleipen dinamikoak egiteko.
arr_iz*[*index*]=*bal	*arr_iz*-aren *index* elementuari *bal* esleipenetu
${*arr_iz*[*index*]}	*arr_iz*-aren tauleko *index*-an dagoen elementu bati erreferentzia egin.

Hurrengo hedapenekin taularen elementu eta ezaugarrei erreferentzia egin diezakegu:

HEDAPENAK	
${*arr_iz*[#]}	taularen elementu kopurua
${*arr_iz*[@]}	taula baten elementuak zerrendatu, bakoitza kate bat bezala hartuz
${*arr_iz*[*]}	kate bakarra bezala hartutako taularen elementuak zerrendatu
${#*arr_iz*[*index*]}	${*arr_iz*[*index*]}-aren luzera

::adibidea::

'deskripzioa' eta 'komandua'-z osatutako bikotea argumentu bezala onartzen duen script-a (gen_menu.sh) zeinek aukera menu bat sortzen du non edozein komandu exekutatu daiteke dagokion deskripzioa aukeratuz.

3.2. funtzioak

Funtzio baten definizioa: Bi aukera ditugu funtzio bat definitzeko:

```
funtzio_iz(){
   ...
   instrukzioak
   ...
}
```

```
function funtzio_iz{
   ...
   instrukzioak
   ...
}
```

Parametroak: funtziora iritsitako parametroei erreferentzia egiteko, 2.4. atalean azaldutako eran egingo dugu ($1, ..., $n,...). Parametro guztiak balio bezala pasatzen dira, hau da, funtziotik bueltatzerakon parametroen balioa ez da aldatuko.

Itzulera: Funtzioak balio bat itzultzea egin dezakegu.

ITZULERA	
return *balioa*	funtzioaren itzulera kodeari balio bat eraspenetuz funtzioa etetzen du.

::adibideak::

1::: (menu.sh) desberdin aukera dituen menua

2::: (urteb_sor.sh) "urtebetetze.txt" hurrengo formatua duten lerroak izendatuko fitxategi bat daukagu:

izena:urtebetetzeeguna

"urtebsor" izendatutako script-a idazten dugu zeinek crontab komanduari pasatutako fitxategia sortzen duena eta urtebetetze bakoitzaren aurreneko egunean mail bat bidaliko lizukeena hurrengo informazioarekin:

```
...
Subject: Urtebetetze oroigarria
...

Bihar [data], [izena]-ren urtebetetzea da. Ez ahaztu zoriontzea.
```

3.3. adierazpen erregularrak

Adierazpen erregularrak: testu eta maisu baten artean dauden hurbiltasunak bilatzeko tresna bat da. Hemen emandako azalpena Perl estiloko adierazpen erregularretan oinarritua dago.

KOMODINA	
.	edozein karaktere
KARAKTERE KLASEAK	
[*lista*]	listaren zenbait karaktere
[*min-max*]	*min eta max*-en artean dauden karaktereak
[^*lista*]	listan ez dagoen edozein karaktere
\w	hitza
\W	\w-ren kontrakoa
\s	hutsuneak, tabuladoreak,... (hutsune, \t, \n, \r)
\S	\s-ren kontrakoa
\d	dijitua
\D	\d-ren kontrakoa
\A	katearen hasieratik begiratzen hastea

\Z	katearen bukaeratik begiratzen hastea
\b	hitzaren mugekin bat etorri
\B	\b-ren kontrakoa
[[:alnum:]]	alfazenbakizkoak
[[:alpha:]]	alfabetikoak
[[:cntrl:]]	kontrol karaktereak
[[:digit:]]	dijitoak
KARAKTERE KLASEAK	
[[:graph:]]	grafikoak
[[:lower:]]	minuskulak
[[:print:]]	karaktere inprimagarriak
[[:punct:]]	puntuazio karaktereak
[[:space:]]	hutsuneak
[[:upper:]]	maiuskulak
[[:xdigit:]]	hamaseitar digituak
ALTERNATIBAK	
alter1*|*alter2	*alter1 edo alter2 agertu daiteke*
ZENBATZAILEAK	
?	aurreko adierazpenarekin 0 edo 1 agerpen
*	aurreko adierazpenarekin 0 edo agerpen gehiago
+	aurreko adierazpenarekin agerpen 1 edo gehiago
{*m*}	aurreko adierazpenarekin m agerpen
{*m*,}	aurreko adierazpenarekin m agerpen edo gehiago
{*m*,*n*}	aurreko adierazpenarekin m-tik n agerpenetara
??, *?, +?, {}?	(no 'greedy') berdina baina ez da karaktere gehien hartzen saiatzen (ez 'greedy')
AINGURAK	
^*expr*	*expr-ekin hasitako edozein kate*
***expr*$**	*expr-ekin bukatutako edozein kate*
TALDEAK ETA ERREFERENTZIAK	
(*expr*)	talde bat egin geroago erreferentziatzeko
n	n-garren taldeari erreferentzia
(?:*expr*)	erreferentzia gabeko taldea

:::adibideak:::

1::: [a-z] letra minuskulak
[A-Z] letra maiuskulak
[0-9] zenbakiak
[,'¿!¡;:\.\?] puntuazio karaktereak
[A-Za-z] alfabetuaren letrak
[a-Z] aurrekoaren berdina
[A-Za-z0-9] alfazenbaki karaktereak
[^a-z] dena letra minuskulak ezik
[^0-9] dena zenbakiak ezik

```
2::: a.b          axb aab abb aSb a#b ...
     a..b         axxb aaab abbb a4$b ...
     [abc]        a b c
     [aA]         a A
     [aA][bB]     ab Ab aB AB
     [0-9][0-9][0-9]     000 001 009 010 019 100 999
     [0-9]*       kate_hutsa 0 1 9 00 99 123 456 999 9999
     [0-9][0-9]*          0 1 9 00 99 123 456 999 9999 99999 99999999
     ^.*$         edozein lerro osorik
     [0-9]+       0 1 9 00 99 123 456 999 9999 99999 99999999 ...
     [0-9]?       kate_hutsa 0 1 2 3 ...
     ^a|b         a b
     (ab)*        kate_hutsa ab abab ababab ...
     ^[0-9]?b     b 0b 1b 2b 9b
     ([0-9]+ab)*kate_hutsa 1234ab 9ab9ab9ab 9876543210ab
                          99ab99ab

3::: ([0-9]) \1 ([0-9]) 3 3 5
     el cargol|gat       el cargol el gat
     ho{1,4}la           hola hoola hooola hoooola
     (slashdot|barrapunto)       slashdot barrapunto
     s[aeou]c            sac, sec, soc, suc
     -?[0-9]+            64, -2, 65536, -512
     0x[0-9A-F]{2}                0xF0, 0x3B, 0xAA
     go{2,5}ggle                  google, gooogle, goooogle, gooooogle
     \bkernel\b          kernel, pero no kernelpanic o microkernel
     \d+\.\d+\.\d+\.\d+ 192.168.0.1 (direcció ip)
     (?:\d+\.){3}\d      192.168.0.1 (direcció ip)
     (?:[\w-]+\.)+(?:net|com|org)        ii.jocs.fractals.net
     s/<i>(.*?)<\/i>/<b>\1<\/b>/g        itàliques   =>   negretes   (en
     document html)
     s/(\d+)\.(\d+)\.(\d+)\.(\d)+/\4.\3.\2.\1/  192.168.0.1 =>
                                                1.0.168.192
     (\d+\.\d+\.\d+\.\d+)(?:\n)*\1(?:\n)*\1     3 bider ip berdina
      (\d+\.\d+\.\d+\.\d+\n*){3}                aurrekoa bezala
```

3.4. komandu oso baliagarriak

3.4.1. sed

ez elkarreragile editore bat da, stdin edo fitxategi batetik jasotzen du sarrera, beharrezko eragiketak egiten ditu eta stdout-etik ateratzen ditu emaitzak. Bere funtzionalitate guztietatik 3 boteretsuenak zehaztuko ditugu: idatzi, ezabatu eta ordeztu. Sintaxia horrelakoa da:

sed [*aukerak*] {*operadore* | *script fitxategia*}

AUKERAK	
-r	adierazpen konplexuak erabil izateko. Adibidez, taldeei erreferentzia egin izateko: \n erabil dezakegu n-garren taldeari erreferentzia egiteko, non talde bakoitza (expr)-en bidez mugatzen da.
-f	operadore baten ordez script bat erabili nahi badugu
-n	lanean dagoen lerroa ez du pantailaratzen

OINARRIZKO OPERADOREAK	
barrutip	*barruti*-ren lerroak imprimitzen ditu
barrutid	*barruti*-ren lerroak ezabatzen ditu
s/*maisu1*/*maisu2*/[aldatzailea*]*	*maisu1*-en aurreneko agerpena *maisu2*-renagatik ordezkatzen du
barruti*/s/maisu*1*/maisu*2*/[aldatzailea*]	*barruti*-ren lerroen *maisu1*-en aurreneko agerpena *maisu2*-renagatik ordezkatzen du

ALDATZAILEAK	
g	lerroen agerpen guztiak ordezkatzen ditu, ez bakarrik lehenengoa
i	minuskula eta maiuskulen artean ez du desberdintzen

:::adibideak:::

8d	8. lerroa ezabatzen du
/^$/d	Hutsik dauden lerro guztiak ezabatzen ditu
/kernel/p	kernel hitza duten lerroak imprimatu (-n -rekin)
s/Windows/Linux/	Lerro bakoitzeko “Windows”-en lehenengo instantzia “Linux”-gatik ordezkatu.
s/Windows/Linux/g	“Windows” instantzia bakoitzeko “Linux” ordezkatu
s/ *$//	Lerro bakoitzaren bukaerako zuriuneak ezabatzen ditu
s/00*/0/g	0 sekuentzia guztiak 0 bakar batean murrizten ditu
/Windows/d	Windows agertzen den lerro guztiak ezabatzen ditu
s/Windows//g	Windows hitz guztiak ezabatzen ditu, lerroa ezabatu gabe

:::adibideak:::

```
sed -r 's/(La|El) (.*) és.*/\2/'
```

Subjektua erauzi (artikulua gabe) ondorengo esaldi moetatik: *“La pilota petita és d'un color molt estrany”*

```
sed -r 's/(La|El) ([^ ]*) (.*) (és.*)/\1 \3 \2 \4/'
```

Subjektuaren aurreneko hitza honen ondoren jartzen du.

3.4.2. awk

Testuan oinarritutako datu-baseak prozesatzeko diseinatua dagoen programazio lengoai bat da, nahiz fitxategi zein datu fluxuak izan.

Programa bat eta fitxategi bat, edo gehiagoz, osatzen da awk-ren deia.

awk [-F *bereizle*] [-v *var=balioa*] [-f *prog_awk*|'*prog_awk*'] [*fitxategiak*]

AUKERAK	
-F	lehenetsitako eremu bereizlea
-v var=*balioa*	aldagaiei hasierako balioak definitu programaren exekuzioa izan aurretik
-f *prog_awk*	exekutatu behar dugun programa duen fitxategia

awk programa bat hurrengo sintaxia duen egitura bat edo gehiagoz osatuta dago:

adierazpen { ekintza*1*; *ekintza2*; ... }

Adierazpena: adierazpen logiko edo erregularra.

Ekintzak: Adierazpenaren ebaluazioa positiboa bada adierazitako ekintzak exekutatzen ditu

EKINTZAK	
print	Pantailatik imprimatzen du

:::adibidea:::

```
$ echo -e "\nun\ndos\ntres" | awk '{ print "Jo uso"; print "Linux" }'
Jo uso
Linux
Jo uso
Linux
Jo uso
Linux
```

Lerro bakoitzeko, adierazpena betetzen denez, egiten diren ekintak:
"Jo uso" imprimatu eta "Linux" imprimatu.

Txantiloiak (plantila): Adierazpenaren ordez, txantiloi batzuk jar ditzazkegu zeinek akzio bat edo gehiago exekutatu dezakete programa bukatu aurretik, ondoren edo bietan:

AINTZINSOLAS ETA GIBELSOLAS-EN TXANTILOIAK	
BEGIN	Ekintza hasieran exekutatuko da
END	Ekintza bukaeran exekutatuko da

Beraz, awk programa baten sintaxia horrela izango da:

```
BEGIN {hasierako_ekintzak}
adierazpen1 {ekintza}
adierazpen2 {ekintza}
...
END {bukaerako_ekintzak}
```

Eremuak: awk-ri pasatzen diogun fitxategiaren lerro bakoitza eremuz osatuta dago (mugatzaile bat ez pasatuz gero, zuriune txuria izango da)

EREMUAK	
$*n*	n-garren eremua

:::adibideak:::

1::: Eremuak inprimatu:

```
$ echo “un dos tres” | awk '{ print $1,$2 }'
un dos
```

Zuriune batez bereizirik, lehenengo eta bigarren eremua inprimatzen dugu

2::: Linux hitza agertzen diren lerroak fitxategi batetik erauzi:

```
$ awk '/linux/' fitxategi.txt
```

3::: .new luzapena duten fitxategiei .berri luzapena jarri

```
$ ls -l *.new | awk '{print "mv "$8" "$8".berri"}' | bash
```

4::: Direktorio guztiak ezabatu baina ez fitxategiak:

```
$ ls -l | grep '^d' | awk '{print "rm -r "$8}' | bash
```

5::: Fitxategi guztiak ezabatu baina ez direktorioak:

```
$ ls -l | grep -v ^d | awk '{print "rm "$8}' | bash
```

6::: Zenbaki positiboen batuketa:

batuketaawk.txt:

```
BEGIN { total=0 }
```

0 balioarekin total aldagaia hasieratu

```
$1>0{ total=total+$1 }
```

zenbakia positiboa bada lehenengo eremua batzen dugu

```
END {print "El total és",total}
```

Azkeneko emaitza inprimatzen dugu

```
$ echo -e "1\n2\n-4\n4" | batuketaawk -f awk.txt
El total és 7
```

Aldagaiak: Beste edozein programazio lengoai bat bezala, aldagaiak definitu ditzazkegu.

awk aurre-definitutako aldagai batzuk ditugu ere:

AURRE-DEFINITUTAKO ALDAGAIAK	
FS	eremu bereizlearen karakterea dauka (zuriune txuria lehenetsita)
RS	erregistro bakoitza non bukatzen den esaten duen karakterea dauka (\n lehenetsita)
OFS	sortutako irteerarako eremu bereizlea dauka (zuriune txuria lehenetsita)
ORS	sortutako irteerarako erregistro bukaera zein den esaten duen karakterea dauka (\n lehenetsita)
NF	prozesatzen ari den erregistroaren eremu kopurua dauka
NR	prozesatzen ari den erregistroaren ordena kopurua dauka

:::adibideak:::

1:::

```
$ echo –e "GNU/Linux sistema \neragilea erabiltzen dut" |
awk –v OFS="\t" –v ORS=“\t” ‘{print $1, $2, $3}'
GNU/Linux sistema \neragilea erabiltzen dut
```

Irteera bezala, eremu eta erregistroen bereizketa tabuladore bat da.

2::: awk.txt:

```
{ print "Lerroa prozesatzen", NR }
```

prozesatzen ari garen lerro zenbakia inprimitzen dugu

```
NF > 1 { print "Lerroak hitz bat baino gehiago du" }
```

begiratzen dugu ea eremu bat baino gehiago duen

```
$1 == "GNU/Linux" {print "Lehenengo hitza GNU/Linux da"}
```

lehenengo eremua ea GNU/Linux den begiratzen dugu

```
$ echo -e "GNU/Linux SE bat da \nAtsegin dut
 GNU/Linux\nGNU/Linux nere SE da" | awk -f awk.txt
1go lerroa prozesatzen
Lerroak hitz bat baino gehiago du
Lehenengo hitza GNU/Linux da
2. lerroa prozesatzen
Lerroak hitz bat baino gehiago du
```

Operadoreak: Hurrengo operadoreak ditugu lan egiteko:

OPERADOREAK			
zenbakizkoak		**konparazioa**	
+	batuketa	**>**	handiago
-	kenketa	**>=**	handiago edo berdin
*	biderketa	**<**	txikiago
/	zatiketa	**>=**	txikiago edo berdin
%	hondarra	**==**	berdin
^	berreketa	**!=**	desberdin
ald++	*ald*-1 bat geitu		
ald--	*ald*-i bat kendu		
logikoak		**kateamendua**	
&&	eta	**zuriunea**	kateamendua
||	edo		
!	ezeztapena		

Kontrol egiturak: Edozein programazio lengoaiak duen kontrol egiturak ditu. awk kasuan, C programazio lengoaiatik hartuta dago sintaxia.

if EGITURA	for EGITURA	while EGITURA	EST. do/while

if(*bald*) {
  instrukzioak
}
[else {
  instrukzioak
}]
	for (hasi;bald;instr){
	  instrukzioak
	}
		while(bald){
		  instrukzioak
		}
			do{
			  instrukzioak
			} *while(bald)*

:::adibideak:::

1::: posnegawk.txt:

```
{
   if( $1 > 0 ){            aurreneko eremua positiboa ala

      print $1,"Positibo da";
   }
   else{                    ... negatiboa den
      print $1,"Negatibo da";
   }                        begiratzen dugu...
}

$ echo -e "12\n-15\n10" | awk -f posnegawk.txt
12 positibo da
-15 negatibo da
10 positibo da
```

2::: taulaawk.txt:

```
{
   print "\n",$1, “-ren taula.”
   print "-----------"
   for( i = 1; i < 11; i ++ ){
      j = $1 * i;
      print i,"*",$1,"=",j;
   }
}

$ echo -e "2\n3" | awk -f awk.txt

2-ren taula
-----------
1 * 2 = 2
2 * 2 = 4
3 * 2 = 6
4 * 2 = 8
5 * 2 = 10
6 * 2 = 12
7 * 2 = 14
8 * 2 = 16
9 * 2 = 18
10 * 2 = 20

3-ren taula
-------------
1 * 3 = 3
2 * 3 = 6
```

```
3 * 3 = 9
4 * 3 = 12
5 * 3 = 15
6 * 3 = 18
7 * 3 = 21
8 * 3 = 24
9 * 3 = 27
10 * 3 = 30
```

Funtzioak: Aurre-definitutako zenbait funtzio ditugu ere:

FUNTZIOAK	
length(*s*)	s-ren luzera byte-tan itzultzen du
rand()	0 eta 1 artean ausazko zenbaki bat itzultzen du
srand()	Ausazko zenbakia sortzeko hazia hasten du.
int(*ald*)	*ald* zenbaki oso batean bihurtuta itzultzen du
substr(*s*,*m*,*n*)	*s*-ren azpikatea itzultzen du, *m* posiziotik hasita eta *n* luzerarekin
index(*s*,*t*)	*s*-ren posizioa non *t* agertzen delarik edo 0 ez badago
match(*s*,*r*)	s-ren posizioa non *r* adierazpena betetzen delarik
split(*s*,*a*,*fs*)	*fs*-gatik bereizirik dauden elementuak *a* taulara itzultzen ditu
sub(*r*,*t*,*s*)	*s*-n *t* katea *r*-gatik aldatzen du
gsub(*r*,*t*,*s*)	*s*-n *t* katea *r*-gatik aldatzen du agerpen guztietan
sprintf(*f*,*e*,*e*,...)	Formatuarekin inprimatzen du
system(*cmd*)	cmd exekutatzen du eta itzul-kodea itzultzen du
tolower(*s*)	*s* minuskuletan itzultzen du
toupper(*s*)	*s* maiuskuletan itzultzen du
getline	Fitxategiaren irakurketa bat bortxatzen du

:::adibideak:::

1::: 72 karaktere baino gehiago duten lerroak inprimatzen duen programa:

```
length>72
```

2::: Kaixo hitza, agur ordez jarri

```
{sub("kaixo","agur");print}
```

3::: Lerroak alderantziz inprimatu

```
{for (i=NF; i>0; i--)
   print $i}
```

4::: Fitxategi baten lerroak inprimatu ondoz ondoko errepikatutakoak ezabatuz

```
$1 != aurrekoa {print; aurrekoa=$1}
```

5::: ASCII kodea inprimatu

```
{ for (i=30; i<255; i++)
   printf("karaktere %c\t%d\t%o\n", i, i, i) }
```

6::: Erabiltzaileak bere pasahitzarekin erantsi (beste fitxategi batean zuriune batez bereizirik daudelarik)

```
{ system ("useradd -p `mkpasswd ”$2”` "$1) }
```

3.5. kolore eta mugimendu pixka bat...

Sistema eragilearen azpian desberdin lan egin ahal izateko, American National Standards Institute (*ANSI*)-ak karaktereekin zenbait sekuentzia hornitzen ditu. Orain, ikusiko ditugun sekuentziak, pantailako karaktereei formatua eman (lodia, kolorea,...), kurtsorea mugitu... lortzeko jarri behar duguna:

IHES_SEKUENTZIA+KONTRO_SEKUENTZIA

IHES SEKUENTZIAK	
^[	Testu direktu bat egiten badugu eta cat-ekin ikusteko VI editorearekin karaktere hau lortzeko: ctrl+x ESC
\E	echo -e “” bat egiten badugu

KONTROL SEKUENTZIAK	
TESTUAREN ATRIBUTUAK	
[0m	Testu normala (reset)
[1m	Lodia
[3m	Etzana
[4m	Azpimarratuta
[5m	Keinuka
[7m	Alderantzia
[22m	Ez-lodia
[23m	Ez-etzana
[24m	Ez-azpimarratuta
[25m	Ez-keinuka
[27m	Ez-alderantzia
KOLOREAK	
[XX;YYm	XX: letraren kolorea YY: atzekaldearen kolorea **LETREN KOLOREAK** / **ATZEKALDEAREN KOLOREA** **30** beltza — **40** beltza **31** gorria — **41** gorria **32** berdea — **42** berdea **33** horia — **43** horia **34** urdina — **44** urdina **35** magenta — **45** magenta **36** zian — **46** zian **37** txuria — **47** txuria Balio hauek 0 (testu normala) eta 1-rekin (lodia) konbinatu ezkero kolore indartsuak edo argiak lortzen ditugu
KURTSOREAREN MUGIMENDUA	
[*x*A	*x* lerro igo
[*x*B	*x* lerro jeitsi
[*y*C	*y* zuriune eskubira mugitu
[*y*D	*y* zuriune ezkerrera mugitu
[*y*;*x*H **[*y*;*x*f**	Kurtsorea *y*,*x* posizioan kokatu

[?6h	Goi ezker izkinean kurtsorea kokatu
[s	Kurtsore eta atributuak gorde
[u	Grabatutako atributuak berezartzen ditu
PANTAILAREN KONTROLA	
[2J	Pantaila garbitu
[K	Lerro bukaerarte ezabatu
[?5h	Alderantziz piztu
[?5l	Alderantziz itzali

::adibideak::

1::: Argi berde testua berde azpimarratuaren gainean eta berriz kolore normaletan utzi

```
echo -e “\E[1;4;31;42mKAIXO\E[0m”
```

2::: 15. lerroan eta 40. zutabean atzekalde horia, testu urdina (kolore argiak) eta keinukarekin kurtsorea kokatu. Ondoren kurtsorea 5 lerro beherago kokatu.

```
echo -e "\E[2J\E[15;40H\E[1;5;34;43mKAIXO\E[0m\E[5B"
```

3::: Hamar batuketa proposatzen duen script-a (batuketatesta.sh) eta ateratako nota ematen dizu. Kurtsorearen kolore eta posizio desberdin erabiltzen dira.

4. eranskinak

4.1. ariketak

:::ALDAGAIAK:::

1. Aldagaiak erabiltzen:
   a) “Astelehena” esleitu EGUN1 aldagaiari, “Asteartea” EGUN2 algaiari, EGUN7 aldagaiararte.
   b) Egiaztatzeko dauden aldagai guztien balioak erakutsi.
   c) Aldagai hauek erabiliz hurrengo irteera lortu:
      Astea: Astelehena Asteartea Asteazkena Osteguna Ostirala Larunbata Igandea
   d) Aurre-definitutako aldagaiak soilik erabilitz, ASTEA aldagaian zuriunez bereizita egunen lista kargatu.
2. Inguruneko aldagaiak erabiliz, erakutsi:
   a) erabiltzailearen lan-direktorioaren bidea
   b) erabiltzailearen login izena
   c) erabiltzailea erabiltzen ari den terminalaren izena
   d) uneko komandu interpretearen izena
   e) exekutagarriak bilatzeko bidea/k
   f) inguruneko aldagai guztiak
3. Aldagaiak erabiltzen:
   a) VIES aldagaia “/usr/doc:/var/lib/dpkg” balioarekin erazagupenetu. Bere edukia erakutsi.
   b) VIES aldagaiari /usr/doc/HOWTO direktorioa bukaeran gehitu eta /usr/DOC/FAQ-ren hasieran (direktorio guztiak :-rekin bereiztuz)
4. Zein da hurrengo komanduen irteera?
   ```
   echo $LOGNAME
   echo “$LOGNAME”
   echo '$LOGNAME'
   echo \”$LOGNAME\”
   echo “Nere login-a $LOGNAME da”
   echo 'Nere login-a $LOGNAME da'
   ```
5. Aldagaiak erabiliz:
   a) prompt-a gordetzen den aldagaiaren edukia erakutsi. SAVE izendatutako aldagai lokal batean bere balioa gorde.
   b) prompt-a aldatu horrela azaltzeko:
      UNIX argia$
   c) prompt-a hasierako balioa izatea egin ezazu SAVE aldagaia erabiliz.
6. Aldagaiak erabiliz:
   a) ALD1 aldagaia hasieratu “shell bash 1” katea balioarekin. Bere edukia erakutsi. shell csh deitu. Zein balio du orain ALD1 aldagaiak? Zergatik?
   b) csh-tik atera. Zein balio du orain ALD1-ek? Zergatik?
   c) ALD1 ingurune aldagaia izatea egin eta a) eta b) urratsak errepikatu. Zer gertatu da?
   d) csh deitu ordez bash deitzen badugu, berdina pasatzen da?
7. /usr/prog/bin direktorioa PATH aldagaiari erantsi. Zelan frogatu dezakezu PATH berria ondo funtzionatzen duen?
8. MID aldagaia sortu, HOME eta LOGNAME aldagaien balioa bi

puntukin bereizirik duena.

:::BERBIDERAKETA:::

9. “Lerro fitxategia” testuarekin *lerroak* fitxategia sortu. Sortutako fitxategiari testu lerro bat gehitu, adibidez “Hau lehenengo lerroa da”, edozein testu-editore bat erabili gabe.

  a) cat komanduarekin, /etc/services fitxategiaren edukia pantailaratu
  b) cat komandua idatzi sarrera estandarra /etc/services fitxategitik eta irteera estandarra serveis.txt fitxategira berbideratuz. serveis.txt erakutsi.
  c) echo erabiliz, error.txt fitxategia sortu “Errore fitxategia” edukiarekin.
  d) cat komanduarekin ezexist.xxx fitxategia erakutsi sarrera estandarra berbideratu gabe, baina irteera estandarra ezexist.txt fitxategira berbideratuz eta errore estandarra erroreak.txt fitxategira. ezexist.txt eta erroreak.txt bistaratu.

10. " $ cat f1 f2 " sentetzia idatzi. f1 fitxategia existitzen da eta f2 fitxategia ez da existituko. Pantailatik irteera begiratu.
  a) Errore irteera erroreak izendatutako fitxategi batera bidaltzea lortu.
  b) Existitzen den fitxategiaren edukia f3 izendatutako fitxategi batera kopiatzea lortu (a) atalean lortutakoa aparte).

11. Berbideraketa erabiliz, hurrengo informazioa duen fitxategi bat sortu:

- “Sistemaren txostena” edukia lerro batean
- Bi lerro zuri
- Uneko hilabetearen egutegia
- Uneko data
- Erabiltzen ari garen ordenagailu mota
- Konektaturik dauden erabiltzaileak

:::IRAGAZKI ETA HODIAK:::

12. *text* izendatutako fitxategia emanez:

```
a
aa
ab
aba
aaa
abab
abba
a katea kapikua da
aa katea ere kapikua da
ab katea ez da kapikua
Kate kapikua batetan bere bukaera hasieraren erreflexua
da
Kate eta bere erreflexua konkatenatuz gero emaitza
kapikua da
A
AA
```

ABA

a) 'a' letra duten lerroak zerrendatu.
b) Ondoz ondoko 2 'a' letra duten lerroak zerrendatu.
c) 'a' letra ez duten lerroak zerrendatu.
d) Letra maiuskulak ez duten lerroak zerrendatu.
e) 'a' letrarekin hasitako lerroak zerrendatu.
f) Letra minuskula batekin hasitako lerroak zerrendatu.
g) 'kapikua' katearekin bukatzen diren lerroak zerrendatu.

13. Uneko direktorioaren fitxategi guztiak direktorioak agertu gabe zerrendatu.
14. root UID-a duten sistemaren prozesu guztiak erakutsi.
15. root UID desberdin duten sistemaren prozesu guztiak erakutsi.
16. “who” komanduak emandako emaitzatik erabiltzaileen izenak atera.
17. “who” komanduak emandako emaitzatik 1go eta 3. eremuak atera.
18. $HOME direktorioaren fitxategi guztien baimenak atera.
19. $HOME direktorioaren fitxategi guztien jabe eta luzera zerrendatu.
20. Linux zerbitzarietan /etc/passwd fitxategiak, makinan kontu bat duten erabiltzaile guztien informazioa du. Lerro bakoitzak erabiltzaile bati dagokio eta hurrengo eremuak “:”-z bereizirik agertzen dira:

Erabiltzailearen izena
x forman kodetuta pasahitza
Erabiltzailearen identifikadorea
Erabiltzaileri dagokion taldearen identifikadorea
Erabiltzaileari buruzko informazioa (izena,abizenak,...)
Lan direktorioa
Erabilitako komandu-interpretatzailea

Fitxategi honetan erabiltzaile bati dagokion lerroa ondorengoa bezala izango zen:

maria:x:210:204:Maria Sanchez:/home/maria:/bin/bash

a) Fitxategi honetan zure erabiltzailearen sarrera bat dagoen ala ez, egiaztatu.
b) Fitxategi honetan zure talde berdineko erabiltzaile guztiak zerrendatu.
c) Erabiltzaileen identifikadoreak zerrendatu.
d) Zure talde berdineko erabiltzaileak erabiltzen duten shell-a zerrendatu.
e) Lehenengo eta 3-garrenetik 5-garrenera eremuak zerrendatu.

21. /etc/passwd fitxategiaren edukia alfabetikoki zerrendatuta erakutsi.
22. “who” komanduaren emaitza alfabetikoki zerrendatu.
23. “who” komanduaren emaitzatik jasotako erabiltzaileen izenak atera eta alfabetikoki zerrendatu.
24. “who” komanduaren emaitzatik jasotako erabiltzaileen izenak atera, alfabetikoki zerrendatu eta errepikatuta dauden izenak ezabatu.
25. Uneko direktorioaren fitxategien izen eta luzerak (direktorioak kenduta) luzeraz zerrandutarik zerrendatu.

26. /etc/group fitxategian sisteman dauden taldeak, hauen identifikadoreak eta bere osagaiak zehazten dira. Lerro bakoitzaren formatua horrela da:

talde_izena:talde_id:osag1,osag2,...osagn

a) Zure erabiltzailea dagoen taldeen lerroak pantailaratu.
b) Zenbat taldeetan zauden pantailaratu.
c) Zu eta beste erabiltzaile bat zaudeten taldeen lerroak pantailaratu.
d) Orain, zure erabiltzailea dagokion taldeen izenak soilik pantailaratu.
e) Talde hauen izenak NERETALDEAK izeneko aldagai batean gorde.
f) /etc/passwd fitxategian dagoen zure erabiltzaileari dagokion lerroaren lehenengo eta hirugarren eremua pantailaratu (izena eta identifikadorea). Izena eta identifikadorea ":"-gatik bereizirik egongo da.
g) f) atalean lortutako emaitza atera ezazu, baina orain, izena eta identifikadorea zuriune huts batekin bereiztuta egongo dira.
h) Aurreko irteera NORTASUNA izendatutako aldagai batera esleitu.
i) NI izendatutako aldagai bat sortu, NORTASUNA eta NERETALDEAK aldagaien edukia duena. Emaitza pantailaratu.

27. "Kaixo <num_id>" pantailaratu nahi dugu non num_id zure identifikadorea delarik. Hiru modu desberdinez egin ezazu:
a) id komanduaren aukerak erabiliz
b) id komanduaren aukerak erabili gabe
c) /etc/passwd fitxategia erabiliz

28. *date* komanduaren aukerak gabe erabiliz egin ezazu:
a) Eguna (zenbakia) pantailaratu. EGUNA aldagaian hilabetea gorde.
b) Hilabetea pantailaratu. HILABETEA aldagaian hilabetea gorde.
c) Urtea pantailaratu. URTEA aldagaian urtea gorde.
d) "Gaur <urtea> urtea, <hilabetea> hilabetea eta <egun_zen> eguna da" pantailaratu erabilitako aldagaiekin.
e) gaur.dat fitxategian aurreko lerroa gorde pantailatik atera gabe.
f) gaur.dat fitxategian aurreko lerroa gorde eta pantailatik atera.

:::SHELL-SCRIPTS:::

29. Bi argumentu emanda lehendabizikoa bigarrena baino handiago bada batu, eta ez bada, beraien arteko kenketa egiten duen script bat diseinatu.
30. Karaktere bat eskatu eta zenbaki, letra edo beste gauza bat den esaten digun script bat diseinatu.
31. Pasatuko argumentua direktorio bat bada bere edukia listatzen duen script bat diseinatu.
32. Parametro bezala pasatzen zaizkion zenbaki guztiak batzen dituen script bat diseinatu.
33. Uneko direktorian dauden fitxategien izenen karaktere kopurua kontatzen dituen script bat diseinatu.

34. Parametro bezala pasatzen zaizkion zenbakiak bakoitiak edo bikoitiak diren esaten digun script bat diseinatu.
35. Erabiltzailearen izena eskatuz, existitzen bada esaten duen script bat diseinatu, eta existitu ezkero, konektatua badago esaten duena.
36. Unezko direktoriaren fitxategi eta direktorioak hurrengo moduz pantailaratzen dituen “lse” izendatutako script bat diseinatu:
ls komanduarekin hurrengo irteera jasotzen badugu:

```
fit1   prog1   prog2   joc
```

Gure lse script-arekin horrela azalduko da:

```
|_fit1
.|_prog1
..|_prog2
...|_joc
```

37. Argumentu bezala pasatutako fitxategi guztien luzera batzen dituen “batuluz” izendatutako script bat diseinatu, errore bat emango duena existitzen ez diren edo direktorio diren argumentu bezala pasatutako guztientzat.
38. Argumentu bezala pasatutako direktorioko fitxategi guztien luzera batzen dituen “batudir” izendatutako script bat diseinatu, existitzen ez bada edo direktorio bat ez bada errore mezu bat ematen duena.
39. Fitxategiak kopiatu (-c), mugitu (-m) eta ezabatu (-d) onartzen duen “opf” izendatutako script bat diseinatu. Erabilitako sintaxia zuzena izatea script-ak egiaztatu behar du.
40. Erabiltzaileen izenak eskatzen dizkigun “erab” izendatutako script bat diseinatu, eta hauek existitzen badira eta konektatuak badaude bere konexioaren data eta ordua esango lukena. Erabiltzailea AMAIERA idatzi ondoren bukatuko da script-a.
41. Parametro bezala pasatutako direktorio baten besteak (others) irakurtzeko baimena duten fitxategiak erakusten dituen “lro” izeneko script bat diseinatu.
42. Talde identifikadore bat emanda taldearen erabiltzaile bakoitzerako bere id, bere izena eta bere lan direktorioa listatzen duen “lerab” izendatutako script bat diseinatu. Irteera horrelakoa izango zen:

```
52 TALDEAREN ERABILTZAILEAK(5)
ID. ZEN.      IZENA           LAN DIREKTORIOA
-------------------------------------------------------------
235           maria           /home/maria
246           jordi           /home/jordi
...
```

43. “urtebetetze.txt” izendatutako fitxategiaren ondorengo formatua duten lerroak editatu:
izena:dataurtebetetzea
44. *crontab* komanduari urtebetetzeen aurreko egunean honelako posta bat jasotzeko pasa behar diogun fitxategia sortzen duen “sorurteb” izendatutako script bat idatzi:

```
...
Subject: Urtebetetze oroipena
...
```

Bihar [data], [izena]-ren urtebetetzea da. Zoriontzen ez ahaztu.

45. Hurrengo lanak egiteko crontab-ak diseinatu:
    - Egunero goizeko 7-tan begira dezan /tmp direktorioa zein luzera duen eta 10 Mb baino handiago bada ezabatu eta administratzaileari posta bat bidaltzen diona.
    - Bi egunetan behin, gaueko 12-tan, /home direktorioa trinkotu eta paketatu, eta /usr/src/backups direktorioan gordetzen duena. Direktorioa existitzen ez bada, sortuko du.
    - Egunero, astelehenetik ostiralera, 19:00, 20:00 eta 21:00-tan konektatuta dauden erabiltzailekin txosten bat sortzen duena eta administratzaileari bidaltzen diona. Txosten honek horrelako itxura izango du:

```
DATA: [data]
ORDUA: [ordua]

KONEKTATURIKO ERABILTZAILEAK([erabiltzailekop])
ID.ZEN.       IZENA   TALDA
-------------------------------------------------
[id1]    [erabiz1]    [erabtal1]
[id2]    [erabiz2]    [erabtal2]
[id3]    [erabiz3]    [erabtal3]
...
```

:::ADIERAZPEN ERREGULARRAK:::

46. Horrelako kateak lortzeko behar diren adierazpen erregularrak sortu:
  a) "aba" katea dutenak
  b) 3 “b” jarraian dutenak
  c) 2 “a”-z hasten direnak
  d) “ba”-z bukatzen dena
  e) “a”-tik hasi eta “b”-rekin bukatutakoak (erdian edozer egon daiteke)
  f) “a” soilik izatea (kopurua ez du inporta)
  g) lehenengo “b” bat eta gero zenbait “a”
  h) “a” zein “b” dituena (ordena edo kopurua ez du inporta)
  i) 3 “a” edo 3 “b” jarraian ez duena
  j) “a” eta “b”-ak errepikatu gabe txandatuz
  k) “a” eta “b” bikoteak soilik izatea
  l) “a” edo “b” bakar bat izatea
  m) zenbait “a” eta gero zenbait “b”, edo alderantziz, izatea
  n) “aba” edo “bab” katea izatea
  o) “ba” katea bi bider izatea

47. Adierazpen erregularren bidez hurrengo edukia duten kateak sortu:
  a) zenbaki hamartarrak (hamartar bereizle bezala koma bat erabiliz)
  b) zenbaki hamartarrak (hamartarren bereizle bezala koma edo puntu bat erabiliz)
  c) telefono zenbakiak (bederatzi dijituzkoa eta 9 edo 6-arekin hasten dena)
  d) posta kodeak (bost dijituzkoa eta gehienez, 5-arekin hasten dena)

e) NAN (zazpi edo zortzi dijitu eta letra batez bukatu daitekeena)
f) zenbaki gabeko minuskula hitzak
g) hasierako letra maiuskula soilik duten hitzak
h) hiru edo lau hitz (zenbakirik gabe)

48. /etc/passwd fitxategia erabiliz hurrengoa lortzeko behar diren adierazpen erregularrak idatzi:
a) iruzkin eremuaren hasieran “Unix” hitza duten erabiltzaileak
b) 101 taldearen erabiltzaileak
c) 100, 101 edo 105 taldeen erabiltzaileak
d) digitu bakarreko UID-a duten erabiltzaileak zerrendatu
f) bat edo bi digituko UID-a duten erabiltzaileak zerrendatu
g) izena, zehazki, 4 karaktere duten erabiltzaileak
h) r-z hasitako 4 karaktere duten erabiltzaileen izena

49. Ondorengorako adierazpen erregularrak idatzi:
a) Data (UUUU/HH/EE)
b) Ordua
c) Posta elektronikoa
d) Telefono zenbakia: (93) 841.61.00
e) .org motako url-ak

4.2. proposatutako link-ak

:::shell script-ak:::

http://www.tldp.org/LDP/Bash-Beginners-Guide/Bash-Beginners-Guide.pdf

shell-script-en hasteko kurtsoa, nahiko osoa. Mini-kurtso honetan azaldu gabeko atalak eta aukerak betetzeko.

http://www.ciberdroide.com/misc/novato/curso/

Kontsolaren bidez GNU/Linux-en kurtsoa. shell-scripting-eko atal bat du.

:::adierazpen erregularrak:::

http://bulma.net/body.phtml?nIdNoticia=770

Adierazpen erregularren tutoriala.

http://iie.fing.edu.uy/~vagonbar/unixbas/expreg.htm

Adierazpen erregularren tutoriala.

http://gmckinney.info/resources/regex.pdf

Adierazpen erregularren erreferentzia azkarra.

:::awk:::

http://www.wikilearning.com/awk_paso_a_paso-wkc-31.htm

awk-ri buruz tutorial txikia

http://www.inicia.es/de/chube/Manual_Awk/Manual_Awk_castellano.pdf

awk-ri buruz erdarazko tutoriala

4.3. off-topic

En cap moment no es pot dir *per sempre*. En canvi, sempre hi ha un moment en què cal dir *mai més*. Però: no és el mai més un per sempre? Tanmateix, el per sempre se situa en el temps, mentre el mai més és intemporal.

(Manuel de Pedrolo)