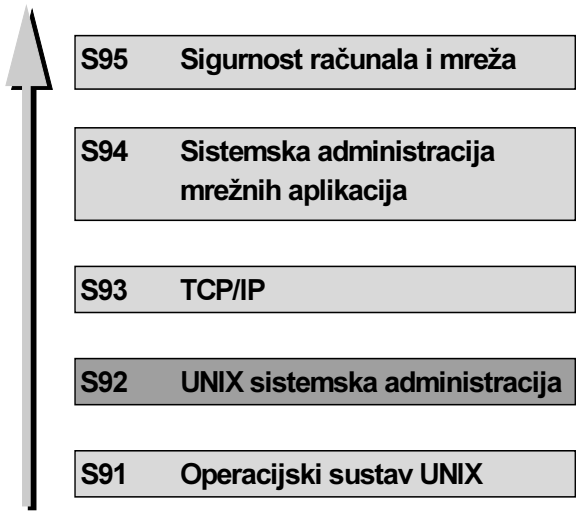
Korn Shell
(sistemska dministracija)

priredio:
Delija Damir

verzija 1.0
Veljača 1998.



Ciljevi tečaja

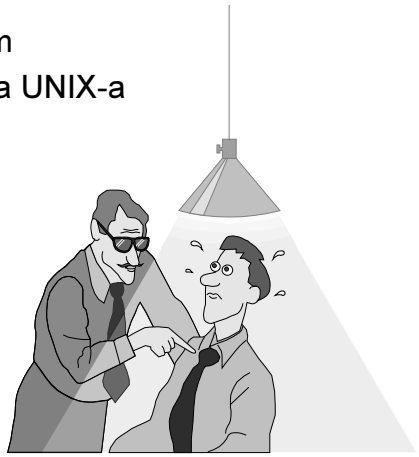
- upoznati se sa upotrebom shell-a
- osposobiti se za samostalno korištenje shell-a

Područja

- Općenito o Korn shell-u
- Pisanje Korn shell skripti

Potrebno predznanje

- Korisnički rad sa UNIX-om
- Poznavanje funkcioniranja UNIX-a



Sadržaj
Dan 1

Uvod	15 min
Osnove korištenja Korn shell-a	30 min
Pauza	
Sintaksa i predefinirane varijable	45 min
Pauza	
Osnovne programske strukture	45 min
Pauza	
Primjeri i vježbe	45 min
Ručak	
Primjeri i vježbe	90 min

Što nećete naučiti na tečaju

- Puno toga!
- Nećete dobiti gotove 'recepte' za sve vaše probleme

Izvori

- Aleen Frisch - Essential System Administration, O'Reilly & Associates, inc.
- Korn shell programming
- OSF1 dokumentacija
- Frank G. Fiamingo, Introduction to Unix System Administration,

Uvod

Cilj:

- upoznati se sa shellom

Unix shell programiranje

- razni shellovi
- ljepilo za vezanje rezluata izvršnih programa
- sistemski konfiguracijski programi
- definiranje korisničkog radnog okruženja

Korištenje shell-a

- Komandni intepreter
- batch jezik
  - "jednostavne komandne skripte"
  - "jednostavni programi"

Historijat

- bourne shell, uz poćetke UNIX-a
- Korn shell naprednija verzija (stalni razvoj)
- C shell napredniji koncept shell-a

- bash, tcsh, zsh

- Namjenski jezici
  - perl, Tcl/Tk, Python ....

Ideja korištenja

- Upotreba vanjskih komandi za izvođenje akcija
- efikasno pokretanje procesa
- efikasno presumjeravanje ulaza / izlaza
- efikasno spremanje rezultata komandi u varijable
- efikasno građenje komandi

```
x=$(ls | grep ps)
```


Nedostaci korištenja

- Složena sintaksa (nerazumljiva za korištenje)
- Veliki overhead kod izvođenja
- Efikasno programiranje zahtjeva poznavanja rada UNIX-a i UNIX utilities!

Komande, I/O redirekcija, pipe

command	izvođenje u foregroundu
command >file	presumjerava stdout u file
command 2>err_file	presumjerava stderr u err_file
command >file 2> file	presumjerava stdout i stderr u file
(command > f1) 2>f2	šalje stdout u f1, stderr u f2
command >>file	dodaje stdout u file

Komande, I/O redirekcija, pipe (2)

command <file	presumjerava stdin iz file
command << text	čita standard input do rijeći “text”
comm1 | comm2	presumjerava stdout iz comm1 u stdin comm2 kroz pipe
command | tee f1 f2	output command poslan u stdout i kopiran u f1, f2
command&	izvođenje u pozadini (background)
nohup command&	background i nakon odjave
set -o monitor	poruka kada background završi

Shell varijable cijeli brojevi

Upozorenje: **=** bez blankova

Integers :

```
n=100 ; x=&n
integer t
typeset -r roues=4   # read only
typeset -i2 x        # x binarni int
typeset -i8 y        # y oktalni int
typeset -i16 z       #
```


Shell varijable stringovi

Strings :

```
lettre="Q" ; mot="elephant”
phrase="Hello, word”
print "n=$n ; lettre=$lettre ; mot=$mot ;
phrase=$phrase"
typeset -r nom="JMB" # string constant
```

Shell varijable polja (arrays)

Arrays : jedno dimenzionalna polja automastki postavljena na 1024 elementa

```
animal[0]="dog" ; animal[1]="horse" ;
animal[3]="donkey”
set -A flower tulip gardenia " " rose
print ${animal[*]}
print ${flower[@]}
print "cell#1 content : ${flower[1]}”
```


Uzorci (pattern matching)

Wild card	matches
?	any single char
[char1char2... charN]	any single char iz the specified list
[!char1char2... charN]	any single char other than one iz the
[char1-charN]	any char between char1 i charN inclusive
[!char1-charN]	any char other than between char1 i charN
|	inclusive
*	any char or any group char (including none)
?(pat1|pat2...|patN)	zero or one the specified patterns
@(pat1|pat2...|patN)	exactly one the specified patterns
*(pat1|pat2...|patN)	zero, one or more the specified patterns
+(pat1|pat2...|patN)	one or more the specified patterns
!(pat1|pat2...|patN)	any pattern except one the specif. patterns

Znak ~

~	home dir (ls ~)
~frenkiel	home dir od **frenkiel**
~+	absulutni path tekućeg direktorija
~-	prethodni direktorij (cd ~-) (or cd -)

Kontrolni znakovi signali

<ctrl c>	Prekida izvođenje procesa(foreground)
<ctrl z>	Suspendira izvođenje procesa(stop)
• bg	šalje u background
• fg	nastavlja u foreground
• kill -option	šalje signal (TERMINATE)
• kill -9 pid	zaustavlja background proces
kill -l :	*ispis svih podržanih signala*
<ctrl d>	End file character

Konfiguracijske datoteke

/etc/profile sytem wide
$HOME/.profile osobno

Koriste se za:
- set & export varijabli
- set options za login shell
- definiranje logoff scripte

Primjer:
```
set -o allexport   #export all varijables
set +o allexport   #turn off allexport feature
```


Konfiguracija (enviroment)

PATH=.:/bin:/usr/bin:$HOME/bin #define command search
CDPATH=.:$HOME:$HOME/games #define search path for cd
FPATH=$HOME/mathlib:/usr/funcs #define path for autoload
PS1='! $PWD> ' #define primary prompt
PS2='Line continues here> ' #define secondary prompt
HISTSIZE=100 #define size history file
TMOUT=0 # KornShell timeout
VISUAL=vi #comm. line editor
ENV=$HOME/.kshrc #pathname env script

- definira aliase i funkcije za interactivno korištenje
- postavlja defaultne opticije za sve pozive ksh
- postavlja varijable koje se koriste u tekućem ksh

Imenovanje Kornshell varijabli

Preporuka !

- lokalne varijable malim slovima
- globalne varijable VELIKIM SLOVIMA

Važno:

KSH ima 3 start-up scripte.
Prve dvije su login scripte (izvode se kod prijave na sistem)
/etc/profile $HOME/.profile
Treća se izvodi kod pokretanja novog ksh-a
$HOME/.kshrc

Rezervirane Kornshell varijable

CDPATH	direktoriji koje cd pretražuje	
COLUMNS	terminal width	80
EDITOR	staza command line editor	/bin/ed
ENV	staza do startup skripte	
ERRNO	error number	
FCEDIT	staza do history file editor	/bin/ed
FPATH	staza do autoload functions	
HISTFILE	staza do history file	$HOME/.sh_history
HISTFILE	nb command history file	128
HOME	login directory	
IFS	set token delimiters	white space
LINENO	broj tekuće linije u skripti	
LINES	terminal height	24
LOGNAME	user name	

Rezervirane Kornshell varijable (2)

MAILPATH	staze do master mail files	
MAIL	staze master mail file	
MAILCHECK	mail checking frequency	600 seconds
OLDPWD	stari PWD	
OPTARG	name argument switch komande	
OPTIND	pozicija opcije u komandnoj liniji	
PATH	command search directories	/bin:/usr/bin
PPID	PID parent	
PS1	command line prompt	$
PS2	prompt više od jedne linije	>
PS3	prompt za 'select'	#?
PS4	debug mode prompt	
PWD	tekući direktorij	
RANDOM	slučajni broj	

Rezervirane Kornshell varijable (3)

REPLY	input repository	
SECONDS	broj sekundi od početka rada	
SHELL	tekući shell (sh, csh, ksh)	
TERM	tip terminala	
VISUAL	command line editor	/bin/ed
$	PID tekućeg shella	
!	PID zadnjeg pozadinskog procesa	
?	return kod zadnje komande	
_	preostali argumenti	

typeset komanda

- typeset -x # exportirane varijable
- typeset -fx # exportirane fukncije

Posebne Kornshell varijable

$1 - $9	pozicioni parametri
$0	ime skripte/komande koja se izvodi
$argv[20]	20ti pozicijski parametara
$#	broj pozicijskih parametara
$?	exit status

Konvencija za exit status:

- *uspjeh* *0*
- *inače* *!= 0*

Posebne Kornshell varijable (2)

$$ PID shella
$! PID pozadinskg procesa
$- opcije za shell
$* svi pozicioni parametri
$@ isto kao $* razlika u interpretiranju:

"$*"	->	"$1 $2 $3"
"$@"	->	"$1" "$2" "$3"

Evaluacija Kornshell varijable

$var vrijednost varijable var
${var} isto kao $var služi za

```
pero=1
echo $pero        -> 1
echo ${pero}s     -> 1s
echo $peros       -> ""
```

Operation	*if str is unset or null*	*else*
var=${str:-expr}	var=expr	var=${string}
var=${str:=expr}	str=expr; var=expr	var=${string}
var=${str:+expr}	var becomes null	var=expr
var=${str:?expr}	expr u stderr	var=${string}

if ... then ... else
if ... then

if koristi exit status **test** komande !

if test
  then
    Komande;
  else *if* test
    Komande;
  fi
fi

Test brojeva

((number1 == number2))
((number1 != number2))
((number1 number2))
((number1 > number2))
((number1 = number2))
((number1 >= number2))

Moguće više načina zapisa, ne znači uvijek isto!

```
if ((x == y))
if test $x -eq $y
if let "$x == $y"
if [ $x -eq $y ]
if [[ $x -eq $y ]]
```

Test stringova

Test raznih objekata: files, directories, links ...

-a object	any type
-f object	regular file, symbolic link
-d object	directory
-c object	character special file
-b object	block special file
-p object	named pipe
-S object	socket
-L object	symbolic (soft) link
-k object	"sticky bit" set

Test stringova (2)

-s object	nije prazno
-r object	readable
-w object	writable
-x object	executable
-O object	ja sam owner
-G object	moja grupa je grupa vlasnika
-u object	set-user-id set
-g object	set-group-id bit set
obj1 -nt obj2	obj1 je noviji obj2
obj1 -ot obj2	obj1 je stariji obj2
obj1 -ef obj2	obj1 isto obj2 (equivalent)

Test stringova (3)

[[string = pattern]]
[[string != pattern]]
[[string1 string2]]
[[string1 > string2]]
[[-z string]] istina ako je dužina od stringa 0
[[-n string]] istina ako je dužina od stringa > 0

Važno: moguće različite sintakse za isti upit

```
if [[ $str1 = $str2 ]]
if [ "$str1" = "$str2" ]
if test "$str1" = "$str2"
```


Inline logički operatori

`cmd1 && cmd2`
if cmd1 TRUE then cmd2

`cmd1 || cmd2`
if cmd1 FALSE then cmd2

kombinacije:
`cmd1 || cmd2 && cmd3`

Matematički operatori

izraz staviti u ““

+	dod.	y = 7 + 10
-	sub.	y = 7 - 10
*	množ.	y = 7 * 4
/	djelk.	y = 37 / 5
%	modulo	y = 37 + 5
>>	shift	y = 2#1011 >> 2
&	i	y = 2#1011 & 2#1100
^	excl ili	y = 2#1011 ^ 2#1100
||	ili	y = 2#1011 | 2#1100

let i expr

- let izvršava numerički izraz u shellu

```
let x=1+2
```

- expr računa izraz kroz podprocese

```
x=`expr 1+2`
```

Kontrola toka

- jump
  - goto
  - break, continue
- return, exit
- loops
  - while
  - for
  - until

Funkcije

```
function IME
{
   commands;
return $val
}
```

Važno: *pozicioni parametri $1 ... $N se odnose na poziv funkcije*

goto

goto my_label

my_label: Komande;

case

```
case value in
     pattern1) command1 ;
               ... ;
              commandN;;
     pattern2) command1 ;
               ... ;
              commandN;;
esac
```

value	value varijabla
pattern	konstanta, pattern ili groupa pattern
command	komanda

while

```
while ( logical expression)
do
   command;
done

#infinite loop : means true
while :
do
       ....
done
```


while posebnosti

```
while read line        # read until an EOF (or <crtl_d> )
do
   .....
done

while read line
do
       ....
done > fname           # redirect input within this
                       # while loop
```

until

```
until( logical expression)
do
      ....
done <fin >fout        # redirect both input and output
```


for

```
for name in  1 2 3 4    # a list   elements
do
    ....
done

for obj in * */*     # $PWD and the next level below
do
    ....
done
```

Više o petljama

- prekida petlju (while, until, for)
  break;
- u preskaće jednu iteraciju
  continue;
- Ugnježdene petlje dozvoljene

select

```
select ident in Un Deux              # a list   identifiers
do
   case $ident in
       Un) ..... ;;
       Deux) ..... ;;
       *) print " Default" ;;
   esac
done
```

Debug mode

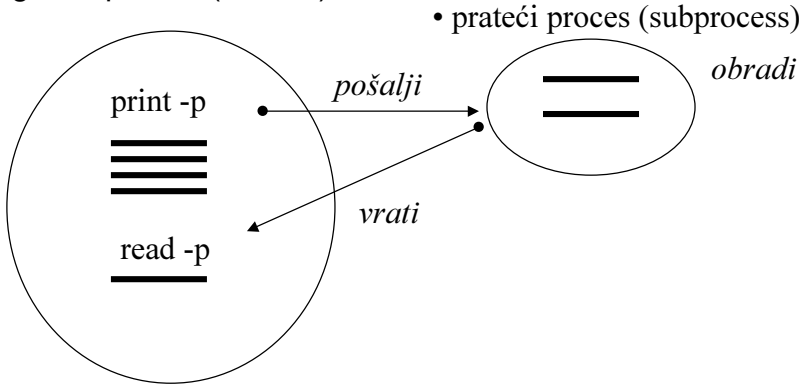
ksh -x script_name

u interaktivnom radu

```
set -x          # start debug mode
set +x          # stop  debug mode
```


Subprocess

- glavni proces (master)
- prateći proces (subprocess)



Subprocess

Subprocess paralelni proces!

```
command |&
```

- print -p šalje line u subprocess
- read -p read line iz subprocess
- line-by-line communication

Osnove Kornshell programiranja

- skripta obavezno
  - postavlja enviroment za procese koji se pozivaju
    - preporuća se potavljenje PATH-a ručno u skripti ili pozivanje komandi punom stazom
    - smešta se u neki sigurni direktorij
    - radi trap komadu za moguće signale
  - brine se za preusmjeravanje ulaza i izlaza
  - čisti mougće smeće iza sebe (trap 0)
  - ima tvz “magic cookie“ na početku

Osnove Kornshell programiranja (2)

- shell skripta samo niz komadi
- shell program koristi i if, petlje, funkcije
- često se brkaju pojmovi

Literatura

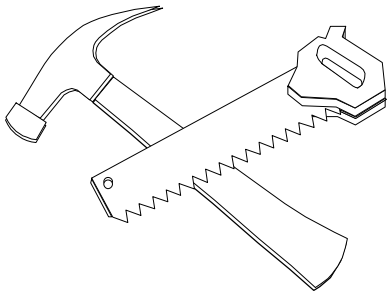
- Aleen Frisch - Essential System Administration, O'Reilly & Associates, inc.
- SunOS dokumentacija
- Frank G. Fiamingo, Introduction to Unix System Administration
- Linux dokumenatcija
- AIX dokumentacija
- Dec Unix dokumentacija
- SAGE dokumentacija

Pitanja

?

Pauza
Ručak

Praktičan rad



Vježba

- Primjeri skripti za backup
- Primjeri skripti za praćenje zauzetosti diska

Primjer skripte za backup ručno pokretanje

```
#!/bin/sh
# A dataless system to a tape drive on a server.
# Script to do a complete backup of the system
echo "*****************************************************************"
echo "This program will allow you to backup GALLIFREY onto magtape"
echo " Follow the directions given below."
echo "*****************************************************************"
echo "Mount tape for partition a and g"
echo " then type RETURN "
read start
echo " ...working - Starting GALLIFREY backup "
#dumps are here
/usr/etc/dump 0ufsdb server:/dev/nrst8 6000 54000 126 /dev/sd0a && echo
   "Done with partition a ..."
/usr/etc/dump 0ufsdb server:/dev/rst8 6000 54000 126 /dev/sd0g && echo
   "Done with partition g ..."
```


Primjer skripte za backup korištenje cron-a

```
#!/bin/sh
HOST='hostname';admin=frank;Mt=/bin/mt
Dump=/usr/etc/dump
device=/dev/nrst0;size=6000;dens=54000;blksz=126;
#Failure - exit
failure () {
/usr/ucb/mail -s "Backup Failure - $HOST" $admin << EOF
    $HOST
    Cron backup script failed. There was no tape in the device.
    EOF
exit 1; }
#Dump Failure - exit
dumpfail () {
    /usr/ucb/mail -s "Backup Failure - $HOST" $admin << EOF
    $HOST
    Cron backup script failed. Could not write to the tape.
    EOF
exit 1; }
```

Primjer skripte za backup korištenje cron-a (2)

```
# Success
success () {
/usr/ucb/mail -s "Backup completed successfully - $HOST" $admin << EOF
$HOST  Cron backup script was apparently successful. The /etc/dumpdates
file is:/bin/cat /etc/dumpdates'
EOF
}
# Confirm that the tape is in the device
$Mt -f $device rewind || failure
for i in blksz /dev/sd0a blksz /dev/sd0g blksz /dev/sd2h
do
   $Dump 0ufsdb $device $size $dens $i || dumpfail
done
($Dump 0ufsdb $device $size $dens $blksz /dev/sd2a || dumpfail) && \
 success
$Mt -f $device rewoffl
```


Pračenje zauzetosti diska

```
#!/bin/ksh
PATH=/bin;/usr/bin
OUTPUT=/dev/console
AWK=nawk

if test "$1" = ""
then
    CMD="iostat -x 30”; HOST=$(hostname)
else
    CMD="rsh $1 exec iostat -x 30”; HOST=$1
fi

exec $CMD | \
$AWK -v host="$HOST" '$1 ~ "sd"{if ($1>30 && $8>50.0)\
printf ( "%s: ", host ); print }' > $OUTPUT
```