



TCSH shell programiranje

4

U TCSH SHELLU MOŽE DA SE PROGRAMIRA, KAO I U BASH SHELLU. MOŽETE DA definišete promenljive i da im dodeljujete vrednosti. Definicije promenljivih i Linux komande možete da postavite u skript fajl, a zatim da ga izvršite. Možete da koristite petlje i upravljačke strukture za ponavljanje Linux komandi ili za donošenje odluka o tome koje će komande da se izvršavaju. Možete da postavite i zamke (traps) u programu tako da se spreči prekidanje izvršenja programa.

TCSH shell se razlikuje od ostalih po tome što se njegove upravljačke strukture malo više uklapaju u format programskih jezika. Na primer, test uslov za upravljačku strukturu TCSH shella je izraz koji vraća vrednost true ili false, a ne Linux komandu. Izrazi u TCSH shellu koriste iste operatore kao i programski jezik C. Možete da izvršavate različita dodeljivanja, aritmetičke, relacione i bitwise operacije. TCSH shell Vam takođe omogućava da deklarišete numeričke vrednosti koje mogu lako da se koriste u ovakvima operacijama.

TCSH shell promenljive, skriptovi i argumenti

TCSH shell koristi shell promenljive na isti način kao i BASH shell. Možete da definišete promenljive u shellu i da im dodeljujete vrednosti, kao i da pozivate argumente skripta. Takođe, možete da definisete promenljive okruženja koje rade na sličan način kao eksportovane promenljive u BASH shellu. TCSH shell se razlikuje po načinu definisanja promenljivih i tipu promenljivih kojeg možete da definišete. TCSH shell definiše promenljive pomoću komandi `set`, `@`; i `setenv`. TCSH shell Vam takođe omogućava da definišete numeričke promenljive i nizove. Komanda `@` definiše numeričku promenljivu na kojoj mogu da se izvršavaju aritmetičke operacije. Male i velike zagrade Vam omogućavaju da definišete i pozivate nizove.

Skriptovi takođe funkcionišu na isti način, ali postoji nekoliko suštinskih razlika. TCSH shell mora da počinje sa tarabicom (#) u prvoj koloni prve linije.

Takođe, iako se prompt može odštampati komandom **echo**, morate da preusmjeravate standardni ulaz u promenljivu.

TCSH shell promenljive

U TCSH shellu morate najpre da deklarišete promenljivu pre nego što počnete da je koristite. Možete da deklarišete promenljivu pomoću komande **set** iza koje sledi ime promenljive. Evo pravila za imenovanje promenljivih:

- Mogući su bilo koji nizovi alfabetских znakova, uključujući karaktere za podvlačenje.
- Mogu da sadrže brojeve, ali broj ne može da bude prvi karakter imena.
- Nisu dozvoljeni ostali tipovi karaktera, kao što su uskličnici, tačka zarez ili blanko znak (razmak). Ovi simboli su rezervisani za shell.

Sledeći primer deklariše promenljivu greeting. Promenljivu možete kasnije da razdefinišete komandom **unset**.

```
set greeting
```

NAPOMENA

Ime ne može da uključuje više reči jer shell razdvaja komandnu liniju blanko znakom. Blanko znak je graničnik za razdvajanje elemenata u komandnoj liniji.

Komandu **set** koristite i za dodeljivanje vrednosti promenljivoj. Unesete ključnu reč set; ime promenljive; operator dodele, **=**; a zatim dodeljenu vrednost. Promenljivoj može da se dodeli bilo koji niz karaktera. U sledećem primeru, promenljivoj greeting je dodeljen string "hello".

```
> set greeting="hello"
```

Kod operacije dodele u TCSH shellu, morate da postavite blanko znak sa obe strane operadora dodele, ili ih uopšte ne navoditi. Ovakav operator dodele

```
> set greeting = "hello"
```

je neispravan jer postoji blanko znak pre operadora dodele (**=**), a posle ga nema.

Možete da dobijete listu definisanih promenljivih korišćenjem komande **set** bez argumenata. Sledeci primer koristi **set** za prikazivanje liste svih definisanih promenljivih i njihovih vrednosti:

```
> set
greeting hello
poet Virgil
```

Kao i u BASH shellu, znak za dolar (**\$**) je specijalni operator koji izračunava shell promenljivu. Izračunavanje vraća vrednost promenljive - obično niz karaktera. Ovaj niz karaktera zamenjuje ime promenljive. Zato, svaki put kada pre imena

promenljive postavite \$, shell zamenjuje ime promenljive njenom vrednošću. U sledećem primeru, izračunava se shell promenljiva greeting, a zatim se njen sadržaj "hello" koristi kao argument u **echo** komandi. Komanda **echo** štampa niz karaktera na ekranu.

```
> echo $greeting
hello
```

Kao i u BASH shellu, jednostruki i dvostruki znaci navoda i kose crte sprečavaju izračunavanje specijalnih karaktera. Takođe, kosi navodnici mogu da se koriste za dodelu rezultata komande promenljivoj. U sledećem primeru, dvostruki znakovi navoda okružuju specijalni karakter ?

```
> set notice = "Is the meeting tomorrow?."
> echo $notice
Is the meeting tomorrow?
>
```

TCSH shell skriptovi: Ulaz i izlaz

Možete lako da definišete korišćenje promenljivih u shell skriptu. Kao u primeru koji će sledeći put da bude naveden, postavljate Linux komande, kao što su operator dodele i **echo**, u fajlu korišćenjem tekstualnog editora. Zatim fajl napravite izvršnim i pozovete ga iz komandne linije kao sledeću komandu. Zapamtite da morate da dodate dozvolu za izvršavanje komandom **chmod** sa u+x dozvolom ili apsolutnom dozvolom 700. Unutar skripta možete da koristite komandu **echo** za štampanje podataka. Međutim, ulaz se čita u promenljivu preusmeravanjem standardnog ulaza sa > operatorom.

NAPOMENA

TCSH shell nema odgovarajuću verziju za **read** komandu iz BASH shella.

TCSH shell ispituje prvi karakter fajla kako bi odredio da li je reč o TCSH shell skriptu. Zapamtite da svi TCSH shell skriptovi moraju u prvoj liniji kao prvi znak da imaju #. Na taj način se identificuje TCSH shell skript. Primetite znak # na početku greet skripta. Kada se # postavi na bilo kom drugom mestu u fajlu, tretira se kao običan karakter.

```
# 
# Script to output hello greeting

set greeting="hello"
echo The value of greeting is $greeting
```

U sledećem primeru se skript *greet* postavlja izvršnim, a zatim izvršava.

```
> chmod u+x greet
```

```
> greet
The value of greeting is hello
```

Komanda **set** u kombinaciji sa operacijom preusmeravanja, \$< čita sve što korisnik unese preko standardnog ulaza. U sledećem primeru se čita korisnički ulaz za promenljivu **greeting**.

```
> set greeting = $<
```

Možete da postavite prompt u istoj liniji koju ulaz koristi za komandu **echo**. TCSH shell ima posebnu opciju za **echo**, opciju **-n**, koja eliminiše potrebu za pritiskanjem tastera za prelazak u novu liniju (carriage return) na kraju ulaznog stringa. Kurzor ostaje u istoj liniji na kraju ulaznog stringa:

```
> echo -n Please enter a greeting:
```

Ako želite da uključite blanko znak na kraju prompta, morate da postavite izlazni string sa dvostrukim znacima navoda, uključujući i blanko znak:

```
> echo -n "Please enter a greeting: "
```

Dole prikazani *greetpt* skript sadrži TCSH shell verziju prompta koji ostaje u istoj liniji kao i ulaz.

```
#  
  
echo -n "Please enter a greeting: "  
set greeting = $<  
  
echo "The greeting you entered was $greeting"
```

Sledi izvršenje *greetpt* skripta:

```
> greetpt
Please enter a greeting: hello
The greeting you entered was hello
>
```

Nizovi: () i

U TCSH shellu možete da deklarišete i koristite nizove, i da pozivate svaki elemenat niza. Nizove deklarišete komandom **set** i listom vrednosti za svaki elemenat niza. Lista vrednosti se navodi u okviru zatvorenih zagrada, a svaka vrednost je odvojena blanko znakom. Veličina ovog niza odgovara broju dodeljenih vrednosti. Sledeći primer deklariše niz pod nazivom **weather** i dodeljene su mu tri vrednosti (pogledajte sliku 4.1). Niz **weather** će da ima tri elementa, gde svaki ima njemu odgovarajuću vrednost:

```
> set weather = (hot cold rain) ←
```

Lista vrednosti

Imenom niza se pozivaju elementi u nizu. Sledeći primer štampa ceo niz.

```
> echo $weather
hot cold rain
```

Pojedinačne elemente niza možete da pozivate postavljanjem elementa unutar zatvorenih velikih zagrada. Elementi u nizu su numerisani, počevši od 1. Sledеći primer dodeljuje novu vrednost prvom elementu niza:

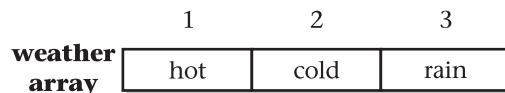
```
> set weather[1] = sunny
> echo $weather
sunny cold rain
```

Indeks niza

Zatim, možete da pristupate sadržaju elementa, baš kao i promenljivoj, tako što ispred elementa navedete znak za dolar (\$). Sledеći primer štampa sadržaj trećeg elementa:

```
> echo ${weather[3]}
rain
```

Izračunava element niza



weather[1] weather[2] weather[3]

% set weather (hot cold rain)

Slika 4.1
TCSH nizovi

U *wreport* skriptu se definiše niz **weather** i koristi u samom skriptu:

```
# 
set weather = (hot cold rain)
echo "The weather today is ${weather[2]}"
echo "Tomorrow there will be ${weather[3]}"
```

Dole je prikazano izvršenje *wreport* skripta:

```
> wreport
The weather today is cold
Tomorrow there will be rain
>
```

Opseg elemenata se može specificiranjem početka i kraja opsega, odvojenih crticom unutar zagrada. Sledеći primer štampa vrednosti drugog i trećeg elementa:

```
> echo ${weather[2-3]}
```

cold rain

Skup elemenata niza

Pitajte stručnjaka

Pitanje: Kako da znam broj elemenata niza?

Odgovor: U svakom nizu postoji promenljiva u kojoj se čuva broj elemenata niza. Promenljiva se poziva navođenjem imena niza ispred kojeg стоји znak #. Na primer, promenljiva koja čuva broj elemenata niza **weather** je **#weather**. Kao i u slučaju bilo koje druge promenljive, sadržaju **#weather** pristupate navođenjem \$ ispred nje, **\$#weather**. U ovom primeru se štampa broj elemenata niza **weather**:

```
> echo $#weather
3
```

Niz argumenata: argv

Kada se poziva shell skript, sve reči u komandnoj liniji se razdvajaju i postavljaju u elemente niza pod nazivom argv. Niz **argv[0]** pamti ime shell skripta, i počevši od **argv[1]**, svaki sledeći element pamti jedan argument unešen u komandnoj liniji. U slučaju shell skripta, **argv[0]** uvek pamti ime shell skripta. Kao i kod bilo kog elementa niza, možete da pristupite sadržaju elemenata niza argumenata navođenjem operatora \$. Na primer, **\$ argv[1]** pristupa sadržaju prvog elementa u nizu argv. U **greetarg** skriptu se pozdrav prosleđuje kao prvi argument u komandnoj liniji. Prvom argumentu se pristupa sa **\$ argv[1]**.

```
#  
echo "The greeting you entered was: $argv[1]"
```

Sledi izvršenje **greetarg** skripta:

```
> greetarg Hello  
The greeting you entered was: Hello
```

Sve reči u komandnoj liniji se razdvajaju, osim ako nisu navedene pod znacima navoda. U sledećem primeru se poziva **greetarg** skript sa stringom bez, a zatim sa znacima navoda. Primetite da se string pod znacima navoda "Hello, how are you" tretira kao jedan argument.

```
> greetarg Hello, how are you  
The greeting you entered was: Hello,  
> greetarg "Hello, how are you"  
The greeting you entered was: Hello, how are you  
>
```

Kada se unese više argumenata, argumenti mogu da se pozivaju pomoću odgovarajućih elemenata u argv nizu. U sledećem primeru, **myargs** skript štampa četiri argumenta. Unešena su četiri argumenta u komandnoj liniji (pogledaj sliku 4.2)

```

#
echo "The first argument is: $argv[1]"
echo "The second argument is: $argv[2]"
echo "The third argument is: $argv[3]"
echo "The fourth argument is: $argv[4]"

```

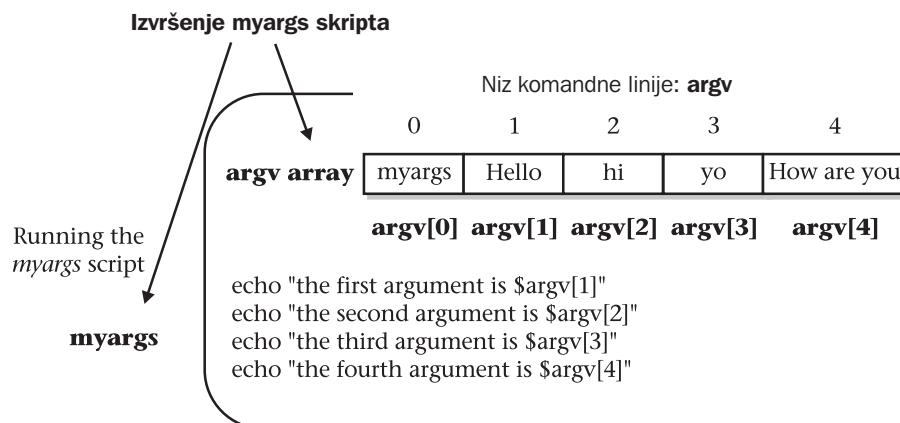
Sledi izvršenje *myargs* skripta:

```

> myargs Hello Hi yo "How are you"
The first argument is: Hello
The second argument is: Hi
The third argument is: yo
The fourth argument is: How are you

```

Skript *wreport* je prepisan kao *wreporta* kako bi koristio argumente. Argumenti *wreporta* skripta su brojevi koji indeksiraju *weather* niz. Prvi argument je 2, kojim se poziva promenljiva `$ argv[1]`. Drugi argument je 3, kojim se poziva `$ argv[2]`. U ovom primeru `$weather[$argv[1]]` poziva *weather[2]*, a to je cold.



Slika 4.2

Niz argv za argumente komandne linije

```

#
set weather = (hot cold rain)
echo "The weather today is $weather[$argv[1]]"
echo "Tomorrow there will be $weather[$argv[2]]"

```

Sledi izvršenje *wreporta* skripta:

```

> wreporta 2 3
The weather today is cold
Tomorrow there will be rain
>

```

SAVET

Element **argv** može da bude skraćenica za broj elemenata kojem prethodi znak **\$**. **\$ argv[1]** može da se piše i kao **\$1**. Ovo znači da TCSH shell može da uključuje pozive argumenata što je veoma slično pozivu argumenata u BASH shellu.

Pitajte stručnjaka

Pitanje: Mogu li da pozovem sve argumente odjednom?

Odgovor: Specijalni argument promenljive **argv[*]** poziva sve promenljive u komandnoj liniji. **\$ argv[*]** može skraćeno da se zapisuje sa **\$***. Setite se da se na isti način pozivaju svi argumenti u BASH shellu. U sledećem primeru, skript **allargs** koristi i **\$ argv[*]** i **\$*** za poziv svih argumenata.

```
#  
echo $argv[*]  
echo $*
```

Sledi izvršenje **allargs** skripta:

```
> allargs Hello Hi Salutations  
Hello Hi Salutations  
Hello Hi Salutations
```

Pitanje: Mogu li da znam broj argumenata koje korisnik unese u komandnoj liniji?

Odgovor: Argument promenljiva **#argv** sadrži broj argumenata koji se unose u komandnoj liniji. Ovo je korisno kada se određuje fiksani broj argumenata za skript. Broj može da se proveri kako bi se utvrdilo da li je korisnik uneo tačan broj argumenata. U sledećem primeru, **argnum** skript štampa broj argumenata koje je korisnik uneo:

```
#  
echo "The number of arguments entered is $#argv"
```

Sledi izvršenje **argnum** skripta.

```
> argnum Hello hi salutations  
The number of arguments entered is 3
```

Arglist skript i slika 4.3 prikazuju korišćenje i **argv[*]** i **#argv** specijalne argument promenljive. Korisnik najpre prikazuje broj argumenata korišćenjem **#argv**, a zatim koristi **argv[*]** kako bi prikazao unešenu listu argumenata.

```
#  
echo "The number of arguments entered is $#argv"  
echo "The list of arguments is: $argv[*]"
```

Sledi izvršenje **arglist** skripta:

```
> arglist Hello hi yo
```

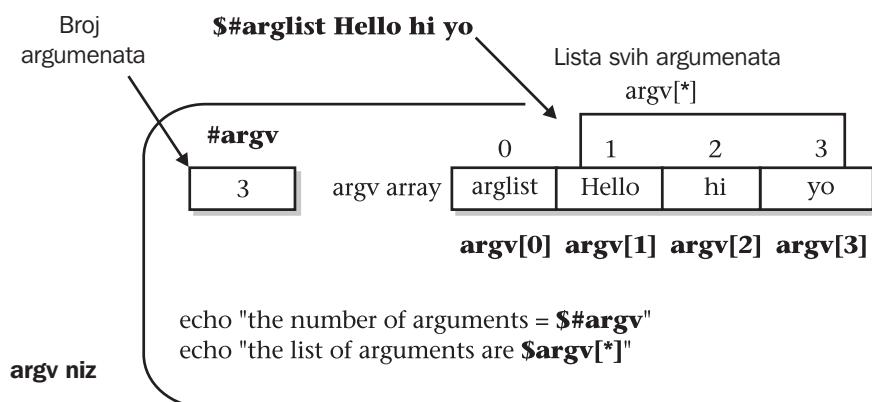
Broj argumenata u komandnoj liniji

Lista argumenata komandne linije

```
The number of arguments entered is 3
The list of arguments is: Hello hi yo
```

Numeričke promenljive: @

U TCSH shellu možete da deklarišete numeričke promenljive korišćenjem @ komande umesto komande set. Zatim možete da izvršavate aritmetičke, relacione i bitske operacije na tim promenljivim (naravno uz odgovarajuće operatore).



Slika 4.3

`#argv i argv[*] promenljive`

Po ovome je TCSH shell sličan programskim jezicima. Numeričke i string promenljive su dva različita tipa objekata, i njima se upravlja na različite načine. Za numeričku promenljivu ne možete da koristite komandu set. Komanda @ sadrži ključnu reč @, ime promenljive i operator dodele, a zatim izraz. U sledećem primeru se deklariše numerička promenljiva **num** i dodeljuje joj se vrednost 10.

```
> @ num = 10
```

Možete da koristite različite operatore dodele, kao što su operatori za inkrement i operatori aritmetičkog dodeljivanja. Isti operatori se koriste i za GAWK (koji će da bude diskutovan u petom poglavlju) i u programskom jeziku C. Izraz može da bude aritmetički, relacioni ili bitski. Možete da kreirate složenije izraze korišćenjem zagrade. Operatori bi trebalo da se u izrazu razdvajaju blanko znakovima; na primer, 10^*5 nije tačan izraz, i trebalo bi da se zapisuje kao $10 * 5$. Takođe, možete da koristite i numeričke promenljive kao operandne u izraza. U sledećem primeru je deklarisana promenljiva **count** kao numerička, a zatim je iskorišćena u aritmetičkom izrazu. Primetite da se **count** izračunava sa operatorom \$ tako da se u izrazu koristi vrednost za **count**, 3.

```
> @ count = 3
> @ num = 2 * ($count +
10)
> echo $num
26
```

Operacija dodele sa aritmetičkim izrazom

TCSH shell aritmetički operatori su navedeni u tabeli 4.1. Sledеći primer koristi dva različita operatora dodele za inkrementiranje promenljive **num** za jedan; operator inkrementa (**++**) i aritmetičko sabiranje sa operatorom dodele (**+=**):

```
> @ num = $num + 1
> @ num += 1
> @ num++
```

Inkrement

Tabela 4.1: TCSH aritmetički operatori i operatori dodele

Aritmetički operatori	Funkcija/Opis
-	Unarni operator minus
+	Sabiranje
-	Oduzimanje
*	Množenje
/	Deljenje
%	Modulo

Operatori dodele	Funkcija/Opis
=	Dodeljivanje
+=	Sabiranje sa izrazom, a zatim dodela
-=	Oduzimanje od izraza, a zatim dodela
*=	Množenje sa izrazom, a zatim dodela
/=	Deljenje izraza, a zatim dodela
++	Inkrementiranje promenljive
-	Dekrementiranje promenljive

Kao što možete da vidite na sledećem primeru, numeričkoj promenljivoj možete da dodeljujete rezultat aritmetičkog izraza.

```
> @ num = 10 * 5
> echo $num
50
```

SAVET

Relacione operacije vraćaju aritmetičku vrednost 1 za true, a 0 za false.

U sledećem primeru, vrednost za **num** će da bude 1 jer je rezultat relacione operacije true.

```
> @ count = 3
> @ num = ($count < 10)
> echo $num
1
>
```

Deklaracija niza numeričkih elemenata je komplikovanija zbog činjenice da Vam je neophodna komanda **set** kako bi se najpre deklarisao niz. Zatim koristite komandu **@** za dodeljivanje numeričkih vrednosti individualnim elementima niza. U sledećem primeru je najpre deklarisan niz **degrees**, a zatim su mu dodeljene vrednosti. Iako ove vrednosti mogu da budu bilo koji brojevi, oni se ne posmatraju kao numeričke vrednosti.

```
> set degrees = (0 45 0)
```

Kada za dodelu numeričke vrednosti elementu koristite komandu **@**, element postaje numerička vrednost. Tada pozivate element navođenjem broja između velikih zagrada. U sledećem primeru se prvom elementu dodeljuje numerička vrednost 100:

```
> @ degrees[1] = 100
> echo $degrees[1]
100
>
```

Promenljive okruženja: **setenv**

TCSH shell ima dva tipa promenljivih: *lokalne* promenljive i *promenljive* okruženja. Lokalna promenljiva je lokalna za shell u kom je deklarisana; promenljiva okruženja funkcioniše slično specijalnoj globalnoj promenljivoj. Promenljiva okruženja je poznata svim subshell-ovima ali ne i roditeljskim shell-ovima. Promenljiva okružnja se definiše komandom **setenv**.

Promenljivoj okruženja možete da dodeljujete vrednosti korišćenjem komande **setenv**, imena promenljive i dodeljene vrednosti. Nema operatora dodele. U sledećem primeru, promenljivoj okruženja **greeting** se dodeljuje vrednost "hello".

```
> setenv greeting hello
```

Kad god se poziva shell skript, on generiše svoj shell. Ako se shell skript izvršava iz nekog drugog shell skripta, imaće svoj shell posebno od shella za prvi skript. Sada postoji dva shella: roditeljski shell koji pripada prvom shell skriptu i subshell, kojeg generiše drugi skript kada se izvršava.

SAVET

Svaki shell ima svoj skup promenljivih. Subshell ne može da poziva lokalne promenljive roditeljskog shella, ali može da poziva promenljive okruženja. Sve promenljive okruženja se deklarišu u roditeljskom shellu i mogu da se pozivaju u svim subshellovima.

U sledećem primeru se promenljiva **myfile** definiše kao promenljiva okruženja za *dispfile* skript. Primetite da se koristi komanda **setenv** umesto **set**. Sada promenljiva **myfile** može da se poziva iz bilo kog subshella, kao što je na primer shell koji se generiše kada se izvršava *prinfile* skript. Korisnik je već kreirao fajl pod nazivom *List* sa ulazima sa ekrana, modema i papira.

Ovde je prikazan *dispfile* skript:

```
#  
setenv myfile "List"  
echo "Displaying $myfile"  
cat -n $myfile  
printfile
```

Sledi *prinfile* skript:

```
#  
echo "Printing $myfile"  
lpr $myfile &
```

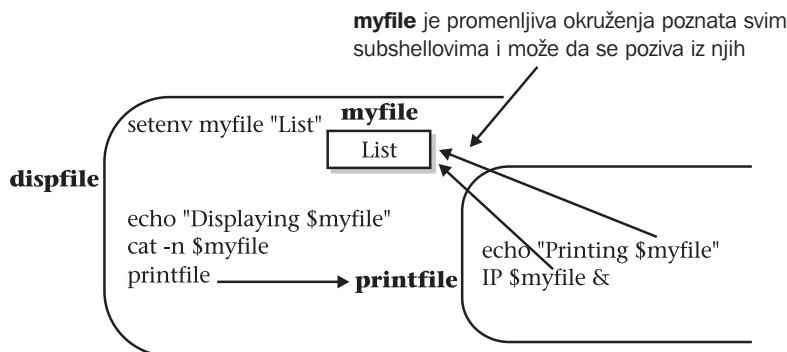
U sledećem primeru je prikazano izvršenje *dispfile* skripta:

```
> dispfile  
Displaying List  
1 screen  
2 modem  
3 paper  
Printing List
```

Kao što je prikazano na slici 4.4, kada se izvršava *prinfile*, moguće je direktno pristupiti promenljivoj **myfile** definisanoj u shellu *dispfile* skripta.

JEDNOMINUTNA VEŽBA

- **Kako se dodeljuju vrednosti promenljivoj?**
 - **Kako ćete promenljivoj da dodelite numeričku vrednost?**
 - **Koji se specijalni niz koristi za čuvanje argumenata shell programa?**
 - *Koristite komandu set i operator dodele.*
 - *Koristite komandu @ i operator dodele.*
 - *Niz argv.*
-



Slika 4.4
TCSH shell promenljiva okruženja pozvana unutar subshella

Tabela 4.2 navodi uobičajene TCSH komande kao i argumente komandne linije

Tabela 4.2: TCSH shell komande i promenljive

TCSH shell komande	Funkcija/Opis
echo	Prikazuje vrednosti.
-n	Eliminiše postavljanje nove linije na izlazu.
eval	Izvršava komandnu liniju.
exec	Izvršava komandu tekućeg procesa. Ne generiše novi subshell, već koristi tekući.
exit	Izlazi iz tekućeg shella.
setenv var	Čini promenljivu dostupnom za pozivanje iz svih novih subshellova (poziv po referenci).
printenv	Prikazuje vrednosti promenljivih okruženja.
set	Dodeljuje novu vrednost promenljivoj. Ako se koristi sama, izlistava sve definisane promenljive.
@	Dodeljuje numerički izraz.
shift	Pomera argument komandne linije za jedno mesto uлево tako да се pozива број увек за један мањи од претходног - на пример, аргумент \$3 се pozива са \$2 и тако редом, \$1 се губи.
unset	Razdefiniše promenljivu.
unsetenv	Razdefiniše promenljivu окруžења.

Argumenti komandne linije	Opis
\$argv[0] \$0	Ime Linux komande.
\$argv[n] \$n	n-ti argument komandne linije koja počinje od 1, \$1 do \$n.
\$argv[*] \$*	Za njihovo menjanje koristite komandu set .
\$#argv \$#	Svi argumenti komandne linije, počevši od 1. Za njihovo menjanje koristite komandu set . Broji argumente komandne linije.

Upravljačke strukture i operatori: **while**, **if**, **switch** i **foreach**

Kao i ostali shellovi, TCSH shell ima skup upravljačkih struktura koje Vam omogućavaju kontrolu izvršenja komandi i skriptova. Možete da koristite petlje i uslovne upravljačke strukture za ponavljanje Linux komandi ili odlučivanje koje će komande da budu izvršene. Upravljačke strukture **while** i **if** su upravljačke strukture opšte namene koje izvršavaju iteracije i donose odluke korišćenjem različitih testova. Upravljačke strukture **switch** i **foreach** su specijalizovanije operacije. Struktura **switch** je ograničeni oblik **if** uslovne strukture koji proverava da li je vrednost jednaka jednoj od vrednosti iz skupa. Struktura **foreach** je ograničeni oblik petlje koji prolazi kroz listu vrednosti, i u svakoj iteraciji promenljivoj dodeljuje novu vrednost.

Petlja **while** i uslovna struktura **if** funkcionišu slično onima u programskim jezicima.

SAVET

TCSH shell se razlikuje od ostalih shellova po tome što su po formatu njegove upravljačke strukture slične onima u programskim jezicima. Uslov koji se ispituje u TCSH shell upravljačkoj strukturi je izraz koji vraća vrednost true ili false, a ne Linux komandu. Ključna razlika između BASH shell i TCSH shell upravljačkih struktura je to što TCSH shell upravljačke strukture ne mogu da preusmeravaju i prosleđuju svoje izlaze. One su striktno upravljačke strukture i kontrolišu samo izvršenje komandi.

Petlja se nastavlja sve dok izraz testa ne vrati false. Uslovna struktura **if** izvršava svoje komande ako njegov izraz testa vraća true. Izraz testa je TCSH shell izraz, a operatori koji se koriste u TCSH shell izrazima su slični onima koji se koriste u programskom jeziku C. Shell koristi široki opseg operatora dodele, aritmetičkih, relacionih i bitskih operatora, s tim da mnoge od njih možete da sretnete i u BASH ili PDKSH shellu. Poput C izraza, TCSH shell izraz je tačan ako vraća nenultu vrednost; netačan je ako je rezultat 0.

Struktura **switch** radi slično ograničenoj verziji **if** strukture. Ona poredi string sa skupom mogućih šablonu. Ako dođe do poklapanja, izvršavaju se operacije dodeljene poklopljenom šablonu. Korisna je za implementaciju menija, kod kojih korisnik bira jednu od nekoliko ponuđenih opcija.

Struktura **foreach** prolazi kroz listu vrednosti i pri tome određenoj promenljivoj dodeljuje svaku vrednost. Nema ispitivanja. Lista vrednosti može da bude sastavljena od šablonu i specijalnih karaktera shella. Na primer, zvezdica (*) generiše listu naziva fajlova. Lista vrednosti može takođe da se odredi i skupom reči. U tom slučaju se svaka reč pridružuje promenljivoj for. Petlja se nastavlja sve dok se ne dodele sve vrednosti.

Test izrazi

Upravljačke strukture **if** i **while** koriste *izraze* u svojim testovima. Tačan test je svaki izraz koji vraća nenultu vrednost. Netačan test je svaki izraz koji vraća vrednost 0. U TCSH shellu kao test izrazi lako mogu da se koriste relacioni izrazi i izrazi jednakosti jer su njihovi rezultati 1 za true, a 0 za false. U izrazima možete da koristite brojne operatore, koje možete da vidite i u tabeli 4.3. Izraz može da bude i aritmetički ili može da poredi stringove, ali stringovi se porede samo da se utvrdi da li su jednakci ili ne.

Za razliku od BASH i PDKSH shellova, u TCSH shellu **if** i **while** test izraze morate da navodite u malim zagradama. Sledeći primer prikazuje test izraz koji ispituje da li su dva stringa jednakci:

```
if ( $greeting == "hi" ) then
    echo Informal Greeting
endif
```

Poređenje jednakosti stringova

TCSH shell ima poseban skup operatora za ispitivanje stringova sa drugim stringovima ili sa regularnim izrazima:

- Operatori **==** i **!=** ispituju jednakost i nejednakost stringova.
- Operatori **=~** i **!~** porede string sa regularnim izrazom i određuju da li je poklapanje šablonu uspešno ili ne.

Tabela 4.3: Operatori test izraza

Poređenje stringova	Funkcija/Opis
==	Jednakost stringova
!=	Nejednakost stringova
=~	Poredi string i šablon radi utvrđivanja jednakosti; šablon može da bude regularan izraz
!~	Poredi string sa šablonom radi utvrđivanja nejednakosti; šablon može da bude regularan izraz
Logičke operacije	
&&	Logičko I
 	Logičko ILI
!	Logičko NE

Testovi za fajlove

-e	Fajl postoji
-r	Fajl se može da čita
-w	Moguć je upis u fajl i modifikacija
-x	Fajl može da se izvršava
-d	Ime fajla je naziv direktorijuma
-f	Fajl je običan fajl
-o	Fajl ima vlasnika
-z	Fajl je prazan

Relacioni operatori

>	Veće nego
<	Manje nego
>=	Veće ili jednako
<=	Manje ili jednako
!=	Različito
==	Jednako

Regularni izraz može da sadrži bilo koji specijalni karakter. U sledećem primeru, sve vrednosti za greeting koje počinju sa malim ili velikim slovom *h* se poklapaju sa regularnim izrazom [Hh]*:

```
if ( $greeting =~ [Hh]* ) then
    echo Informal Greeting
endif
```

Poklapanje sa regularnim izrazom**SAVET**

Kao i BASH shell, i TCSH shell ima nekoliko specijalnih operatora koji ispituju status fajla. Mnogi operatori su isti u oba shella.

U sledećem primeru, komanda if ispituje da li fajl *mydata* može da se čita.

```
if ( -r mydata ) then
    echo Informal Greeting
endif
```

TCSH shell uslovne strukture: if-then, if-then-else i switch

TCSH shell ima skup uslovnih upravljačkih struktura pomoću kojih donosite odluke o tome koje će Linux komande da budu izvršene. Mnoge uslovne upravljačke strukture su slične onima u BASH shellu. Ipak, postoje neke ključne razlike. U TCSH shellu, if struktura se završava ključnom reči *endif*. Struktura **switch** koristi ključnu reč *case* drugaćije nego što se koristi u BASH shellu. Dalje, TCSH shell ima dve if upravljačke strukture: prostu verziju koja izvršava samo jednu komandu i složeniju verziju koja može da izvršava nekoliko komandi, kao zamene drugim komandama. Prosta verzija if strukture se sastoji od ključne reči if iza koje sledi test i jedne Linux komande. Složena verzija se završava ključnom reči *endif*. Uslovne upravljačke strukture TCSH shella su navedene u tabeli 4.4.

struktura if-then

Struktura if-then postavlja uslov za nekoliko Linux komandi. Uslov se daje kao *izraz*. Ako je rezultat izraza različit od nule, izraz je tačan i izvršavaju se komande unutar if strukture.

Tabela 4.4: TCSH uslovne upravljačke strukture

Upravljačke strukture	Opis
if (<i>izraz</i>) then <i>komande</i> endif	Ako je izraz tačan, izvršavaju se komande koje slede. Možete da odredite više Linux komandi.
if (<i>izraz</i>) then <i>komanda</i> else <i>komanda</i> endif	Ako je izraz tačan, izvršava se komanda posle <i>then</i> . Ako izraz nije tačan, izvršava se komanda iza <i>else</i>
switch (<i>string</i>) case <i>šablon</i> : <i>komanda</i> breaksw default: <i>komanda</i> endsw	Omogućava Vam da birate između nekoliko alternativnih komandi.

Ako izraz vraća vrednost 0, izraz nije tačan, a komanda unutar if strukture se ne izvršava.

Struktura if-then počinje sa ključnom reči if iza koje sledi izraz u malim zagradama. Zatim u sledećim linijama odredite Linux komande. Komanda if se završava ključnom reči *endif*. Setite se da je u BASH shellu ključna reč *then* bila u posebnoj liniji, a ovde je u istoj liniji kao i test izraz. Sintaksa za if-then strukturu je:

```
if (izraz) then
    komande
endif
```

Dole prikazani *ifls* skript Vam omogućava izlistavanje fajlova prema njihovoj veličini. Ako u promptu unesete **s**, izlistavaju se svi fajlovi u tekućem direktorijumu, iza kojih sledi broj blokova koje koristi. Ako u prompt unesete bilo šta drugo, **if** test je netačan i skript ne radi ništa.

```
#  
echo -n "Please enter option: "  
set option = $<  
  
if ($option == "s") then  
    echo Listing files by size  
    ls -s  
endif
```

Sledi izvršenje *ifls* skripta:

```
> ifls  
Please enter option: s  
Listing files by size  
total 2  
    1 monday      2 today  
>
```

Često ćete morati da birate između dve alternative zasnovane na tome da li je izraz tačan ili ne. Ključna reč **else** Vam omogućava izbor između dve komande u **if** strukturi. Ako je izraz tačan, izvršavaju se komande koje slede odmah iza ključne reči **then**. Ako izraz nije tačan, izvršavaju se komande koje slede iza ključne reči **else**. Sintaksa za **if-else** komandu je:

```
if (izraz) then  
    komande  
else  
    komande  
endif
```

U sledećem primeru, *elsels* skript izvršava komandu **ls** za izlistavanje fajlova sa dve različite opcije: po veličini i na osnovu informacija fajla. Ako korisnik unese **s**, fajlovi se izlistavaju prema veličini; u suprotnom, izlistavaju se informacije fajla. Primetite da se sintaksa razlikuje od BASH shell verzije *elsels* skripta opisanog u trećem poglavlju.

```
#  
echo Enter s to list file sizes.  
echo otherwise all file information is listed.  
echo -n "Please enter option : "  
set option = $<  
  
if ($option == "s") then  
    ls -s
```

```

else ←
ls -l
endif
echo Goodbye

```

Sledi izvršenje *elsels* skripta:

```

> elsels
Enter s to list file sizes,
otherwise all file information is listed.
Please enter option: s
total 2
    1 monday      2 today
Good-bye
>

```

Pitajte stručnjaka

Pitanje: Kako da koristim if strukturu za proveru argumenata?

Odgovor: Struktura if se često koristi za proveru da li je korisnik uneo odgovarajući broj argumenata za shell skript. Specijalna shell promenljiva **#argv** sadrži broj argumenata koje korisnik unese. Korišćenjem **\$#argv** u test operaciji možete da utvrdite da li je korisnik uneo odgovarajući broj argumenata.

Ako se unese netačan broj argumenata, možda ćete morati da završite shell skript. Kao i u BASH shellu, to možete da uradite pomoću komande **exit**. Komanda **exit** završava shell skript vraćajući izlazno stanje. U **exit** se nalazi broj argumenata: argument 0 označava da je shell skript uspešno završen; bilo koji drugi argument, kao što je 1, ukazuje na pojavu greške

U sledećem primeru, *ifarg* skript uzima samo jedan argument. Ako korisnik ne unese nijedan argument, ili unese više argumenata, **if** test će da bude true i odštampaće se poruka greške, a skript će da vrati pogrešnu vrednost.

```

#
if ( $#argv != 1 ) then ←
    echo "Invalid number of arguments"
    echo "Enter only one argument"
    exit 1
endif

echo $argv[1]

```

Sledi izvršenje *ifarg* skripta:

```

> ifarg
Invalid number of arguments
Enter only one argument

```

Pitanje: Mogu li da se ugnezđavaju if strukture kako bi se donosile složenije odluke?

Odgovor: Za ugnežđavanje **if-then** operacija prosto postavljate **if** strukturu u **else** umesto bloka komandi. Na taj način možete da birate između nekoliko alternativa. Prva alternativa je određena **if** strukturom iza koje slede druge alternative, s tim da je svaka određena svojom **else-if** komponentom. Alternativa za poslednju **if** strukturu je određena sa **else**. Ako je test za prvu **if** strukturu netačan, kontrola se prenosi na sledeću **if** strukturu koja sledi iza sledećeg **else**, i izvršava se njen test. Ako ni on nije tačan, kontrola se prenosi na sledeću **if** strukturu, i izvršava se njen test. Ovo se nastavlja sve dok se ne najde na test koji vraća tačnu vrednost. Tada se izvršava ta **if** komanda, a kontrola se prenosi izvan **if** strukture na sledeću komandu iza ključne reči **endif**. Sledeći primer ilustruje upotrebu ugnežđenih **else-if** operacija.

```
if ($option == "s") then
    ls -s
else if ($option == "l") then
    ls -l
else if ($option == "d") then
    ls -F
else
    echo "Invalid Option"
endif
```

Struktura **switch**

Struktura **switch** pravi izbor između ponuđenih komandi. Slična je **case** strukturi u BASH shellu u smislu da pravi izbor na osnovu poređenja stringa sa nekoliko ponuđenih šabloni. Svakom mogućem šablonu se dodeljuje određeni skup komandi. Ako dođe do poklapanja, izvršavaju se dodeljene komande.

Struktura **switch** počinje ključnom reči **switch** iza koje sledi test string unutar malih zagrada. String se često izvodi iz izračunavanja promenljive. Zatim sledi niz šabloni - svakom šablonu prethodi ključna reč **case**, a završava se dvotačkom. Iza dvotačke se navode komande dodeljene ovom šablonu. Komande se završavaju ključnom reči **breaksw**. Kada se navedu svi šabloni, **switch** struktura se završava ključnom reči **endsw**. Sintaksa za **switch** strukturu je:

```
Switch (test-string)
case sablon:
    komande
    breaksw
case sablon:
    komande
    breaksw
default:
    komande
    breaksw
endsw
```

NAPOMENA

Svaki šablon se upoređuje sa test stringom sve dok se ne pronađe poklapanje. Ako nema poklapanja izvršava se podrazumevana opcija. Podrazumevani izbor se predstavlja ključnom reči default. Ovaj izbor je opcioni. Međutim, veoma je koristan za obaveštavanje korisnika da nije pronađeno poklapanje sa stringom.

Projekat 4-1: Meniji sa switch strukturu

Struktura **switch** se često koristi za implementaciju menija. U sledećem primeru, u programu lschoice, od korisnika se traži da unese opciju za izlistavanje fajlova na različite načine. Primetite da podrazumevana opcija upozorava na neispravan ulaz. Program počinje prikazivanjem opcija menija i zahtevom da korisnik unese jednu opciju. Zatim **switch** struktura detektuje ulaz korisnika i izvršava odgovarajuću komandu.

Korak po korak

1. Kreirajte meni sa tri opcije za izlistavanje prema veličini, prema informacijama fajla ili za C fajlove (fajlovi sa ekstenzijom .c). Koristite tri proste **echo** komande.
2. Zatražite od korisnika da unese izbor, koji se zatim čita u promenljivu **choice** koju koristi komanda **set**, a zatim upotrebite operator \$<.
3. Koristite **switch** strukturu za utvrđivanje izbora koji je korisnik napravio. Izračunajte promenljivu **choice** u **switch** izrazu, **\$choice**, i proverite da li se njena vrednost poklapa sa vrednosti šablonu u case ulazima.
4. Postavite tri case ulaza. Prvi je za šablon **s**, koji ako se poklopi izvršava komandu **ls** sa opcijom **-s**, koja izlistava imena fajlova i njihove veličine. Ulaz se završava komandom **breaksw**.
5. U drugi case ulaz postavite šablon **I**, koji ako se poklopi izlistava fajlove sa svim njihovim informacijama, uključujući datum poslednje modifikacije i dozvole pristupa. Ulaz se završava komandom **breaksw**.
6. Treći ulaz pamti šablon **.c**. U slučaju poklapanja se izvršava komanda **ls.c**, koja prikazuje nazive svih fajlova sa ekstenzijom **.c** (* je specijalni karakter koji se poklapa sa svim šablonima). Ulaz se završava komandom **breaksw**.
7. Za podrazumevani ulaz se prikazuje poruka greške "Invalid Option". Takođe se završava komandom **breaksw**.
8. Struktura switch se završava komandom **endsw**.

Ovde je prikazan *lschoice* skript:

```
# 
echo s. List Sizes
echo l. List All File Information
echo c. List C Files

echo -n "Please enter choice: "
set choice = $<
```

```
switch ($choice)
  case s:
    ls -s
    breaksw
  case l:
    ls -l
    breaksw
  case c:
    ls *.c
    breaksw
  default:
    echo Invalid Option
    breaksw
endsw
```

Sledi izvršenje *lschoice* skripta:

```
> lschoice
s. List Sizes
l. List All File Information
c. List C Files
Please enter choice: c
io.c lib.c main.c
```

U okviru šablona možete da odredite više vrednosti. U prethodnom primeru su dozvoljena samo velika slova. Veliko slovo C bi se smatralo pogrešnim ulazom. Za određivanje više izbora za skup komandi koristite više **case** struktura bez komande **breaksw**. Takođe, možete da koristite specijalne shell karaktere za velike zgrade. U šablonu mogu da se koriste svi specijalni shell karakteri, kao i šabloni u BASH shell **case** strukturi. Pomoću velikih zagrada može da se navede lista mogućih važećih karaktera. Šablon **[Cc]** će da se poklapa i za veliko i za malo slovo c.

U sledećem primeru je *lschoice* ponovo napisan kao *lulschoice*, tako da sada uključuje izbor i za mala i za velika slova. Izbori za unos velikog i malog slova c su navedeni kao dva sukcesivna **case** ulaza bez navođenja komande **breaksw**. Međutim, šabloni za mala i velika slova s su navedeni pomoću uglastih zagrada u jednom šablonu: **[Ss]**.

```
#
echo s. List Sizes
echo l. List All File Information
echo c. List C Files

echo -n "Please enter choice: "
set choice = $<

switch ($choice)
```

```
case [Ss]:  
    ls -s  
    breaksw  
case L:  
case l:  
    ls -l  
    breaksw  
case C:  
case c:  
    ls *.c  
    breaksw  
default  
    echo Invalid Option  
    breaksw  
endsw
```

Sledi izvršenje *lulschoice*:

```
> lulschoice  
s. List Sizes  
1. List All File Information  
c. List C Files  
Please enter choice: C  
main.c lib.c file.c
```

Strukture petlji: while, foreach i repeat

TCSH shell ima skup upravljačkih struktura petlje koje Vam omogućavaju ponavljanje Linux komandi: **while**, **foreach** i **repeat**. TCSH shell upravljačke strukture petlji su prikazane u tabeli 4.5.

while struktura funkcioniše na sličan način kao njoj odgovarajuća struktura u programskim jezicima. Poput if strukture TCSH shella, while struktura ispituje rezultat izraza. **foreach** struktura u TCSH shellu, poput **for** i **for-in** struktura u BASH shellu ne izvršava nikakva ispitivanja. Prosto prolazi kroz listu vrednosti, dodeljujući svaku vrednost određenoj promenljivoj. U tom smislu se foreach struktura značajno razlikuje od odgovarajuće strukture u programskim jezicima. Struktura **repeat** je jednostavna i ograničena upravljačka struktura. Ponavlja jednu komandu određeni broj puta. Ne postoji test izraz, i ne može da ponavlja više od jedne komande.

Struktura while

Petlja **while** ponavlja komande. Ona počinje ključnom reči *while*, a iza nje sledi izraz u malim zagradama.

Tabela 4.5: Upravljačke strukture petlje za TCSH shell

Upravljačke strukture petlji	Opis
while (<i>izraz</i>) <i>komanda</i> end	Izvršava komande dokle god izraz vraća tačnu vrednost.
foreach <i>promenljiva</i> (<i>lista argumenata</i>) <i>komanda</i> end	Ponavlja petlju onoliko puta koliko ima argumenata u listi. Pri svakom prolasku kroz petlju se promenljiva postavlja na sledeći argument u listi; funkcioniše slično for-in strukturi u BASH shellu.
repeat <i>num komanda</i> continue break	Ponavlja komandu određeni broj puta. prelazi na sledeću iteraciju, preskačući ostatak komandi petlje. Prekida izvršenje petlje.

Kraj petlje se određuje ključnom reči *end*. Sintaksa za **while** petlju je:

```
while (izraz)
    komande
end
```

while struktura se lako kombinuje sa strukturom **switch** za prolazak kroz meni. U *lschoice* skriptu možete da uočite da meni sadrži opciju **quit** koja postavlja vrednost promenljive **again** na "ne" i tako zaustavlja petlju.

```
#  
set again=yes  
  
while ($again == yes) ← [Glavna petlja]  
echo "1. List Sizes"  
echo "2. List All File Information"  
echo "3. List C Files"  
echo "4. Quit"  
echo -n "Please enter choice : "  
set choice = $<  
  
switch ($choice) ← [operacija switch za detekciju  
izbora kojeg je uneo korisnik]  
case 1:  
    ls -s  
    breaksw  
case 2:  
    ls -l
```

```

        breaksw
case 3:
    ls *.c
    breaksw
case 4:
    set again = no ← Ovo daje netačnu vrednost testa petlje
    echo Goodbye
    breaksw
default
    echo Invalid Option
endsw

end ← Kraj glavne petlje

```

Sledi izvršenje *lschoicew* skripta:

```

> lschoicew
1. List Sizes
2. List All File Information
3. List C Files
4. Quit
Please enter choice: 3
main.c lib.c file.c
1. List Sizes
2. List All File Information
3. List C Files
4. Quit
Please enter choice: 4
Good-bye
>

```

Struktura foreach

Struktura **foreach** je projektovana za sekvencijalno pozivanje liste vrednosti. Slična je **for-in** strukturi u BASH shellu. Struktura **foreach** uzima dva operanda - promenljivu i listu vrednosti u malim zagradama. Svaka vrednost iz liste se dodeljuje promenljivoj foreach strukture. Kao i **while** struktura, i **foreach** je petlja. Svakim prolaskom kroz petlju se vrši dodata vrednosti iz liste promenljivoj.

Petlja se zaustavlja, kada se dođe do kraja liste. Poput **while** petlje, kraj tela petlje je označen ključnom reči *end*. Sintaksa **foreach** petlje je:

```

foreach promenljiva (lista vrednosti)
    komande
end

```

Foreach petlja je korisna za upravljanje fajlovima. U ovoj strukturi možete kao šablone da koristite specijalne karaktere shella za generisanje liste naziva

fajlova koju koristite kao listu vrednosti. Ova generisana lista fajlova zatim postaje lista koju poziva **foreach** struktura. Zvezdica generiše listu svih fajlova u direktorijumu. Šablon ***.c** izlistava sve fajlove sa ekstenzijom **.c**. To su obično fajlovi sa izvornim kodom na programskom jeziku C.

Sledeći primer pravi kopiju svakog fajla i postavlja kopiju u direktorijum pod nazivom **sourcebak**. Šablon ***.c** generiše listu naziva svih fajlova na kojima funkcioniše **foreach** struktura.

```
#          Dodeljuje sledeći fajl na listu u
          promenljivu backfile u svakoj iteraciji
foreach backfile (*.c) <--> Generiše listu svih fajlova
    cp $backfile sourcebak/$backfile
    echo $backfile
end
```

Sledi izvršenje *cbackup* skripta:

```
> cbackup
io.c
lib.c
main.c
```

Ako se **foreach** struktura navede bez određene liste vrednosti, za listu uzima argumente komandne linije. Kada se pozove shell fajl, argumenti komandne linije postaju lista vrednosti koju poziva **foreach** struktura. Promenljiva koja se koristi u **foreach** strukturi se automatski postavlja na svaku sledeću vrednost argumenta u sekvenci. Prvim prolaskom kroz petlju se promenljiva postavlja na vrednost prvog argumenta. Sledеći put joj se dodeljuje vrednost drugog argumenta i tako redom.

Argument komandne linije možete eksplisitno da poziva korišćenjem argumenta specijalne promenljive **argv[*]**. U sledećem primeru se u komandnoj liniji unosi lista C programa kada se pozove *cbackuparg* shell fajl. U **foreach** petlji **argv[*]** poziva sve argumente komandne linije. Svaki argument se sekvensijalno dodeljuje promenljivoj **backfile** u **foreach** petlji. Prvim prolaskom kroz petlju imamo da je **\$backfile** isto kao i **\$argv[1]**. Sledеćim prolaskom, **\$backfile** postaje jednako **\$argv[2]**. Promenljiva **argnum** se koristi za pozivanje svakog argumenta. I argument i vrednost **backfile** se prikazuju, kako bi se pokazalo da su jednaki.

```
# @ argnum = 1
foreach backfile ($argv[])
    cp $backfile sourcebak/$backfile
    echo "$backfile $argv[$argnum]"
    @ argnum = $argnum + 1
end
```

Sledi izvršenje *cbackuparg* skripta:

```
> cbackuparg main.c lib.c io.c
main.c main.c
lib.c lib.c
io.c io.c
```

JEDNOMINUTNA VEŽBA

- Koja se struktura koristi kao zamena switch strukture?
- Mogu li da se koriste operacije za pokapanje fajlova kao što su *, [] ili ? u testovima?
- Ako se promenljiva koja se koristi u testu petlje nikada ne menja unutar petlje, da li će program ikada da se zaustavi?
 - Ugnežđavanjem if struktura pomoći else.
 - Da.
 - Ne; imate beskonačnu petlju.

Struktura repeat

Struktura **repeat** je struktura koja prosto ponavlja komandu određeni broj puta. Može da ponavlja samo jednu komandu. Struktura **repeat** počinje ključnom reči **repeat**, iza koje sledi broj ponavljanja, a zatim komanda koja će da se ponavlja. Ovo je sintaksa:

```
>repeat num komanda
```

U ovom primeru se komanda **echo** ponavlja tri puta.

```
> repeat 3 echo "hello again"
hello again
hello again
hello again
```

Komanda continue

Kao i u BASH shellu, u TCSH shellu se koristi komanda **continue** sa petljama radi prelaska preko preostalih iskaza u telu petlje radi nastavljanja sledeće iteracije.

NAPOMENA

Korišćenje komande **continue** može da doprinese nejasnoći stila programiranja. Trebalo bi da se ređe koristi.

U sledećem primeru, *loopodd* skript štampa samo neparne iteracije petlje koristeći **continue** za preskakanje **echo** komande kod parnih iteracija. Komanda **continue** uslovno rečeno prenosi kontrolu na ključnu reč **end** za kraj petlje, čime se nastavlja sa sledećom iteracijom petlje.

```
#  
@ num=0  
while ($num < 6)  
    @ num = $num + 1  
    if ( ($num % 2) == 0) continue  
        echo "$num iteration of loop"  
    end  
echo Goodbye
```

Sledi izvršenje *loopodd* skripta:

```
> loopodd  
1 iteration of loop  
3 iteration of loop  
5 iteration of loop  
Goodbye
```

Beskonačne petlje i komanda break

U TCSH shellu možete da implementirate beskonačnu petlju korišćenjem numeričke konstante 1 na mestu test izraza u **while** strukturi. Ako koristite beskonačnu petlju, neophodan Vam je način za njen zaustavljanje. Kada se koristi sa petljom, komanda **break** prenosi kontrolu izvan petlje.

NAPOMENA

Često beskonačna petlja sadrži uslovnu komandu **break** koja kada se izvrši okončava petlju.

U sledećem *loopinfinite* skriptu, korisnik postavlja beskonačnu petlju koja štampa broj iteracije. Kada je **num** veće od 3, izvršava se komanda **break**, i petlja se okončava.

```
#  
@ num=1  
  
while ( 1 )  
    if ($num > 3) break  
        echo "$num iteration of loop"  
    @ num = $num + 1  
    end  
echo Goodbye
```

Sledi izvršenje *loopinfinite* skripta:

```
> loopinfinite  
1 iteration of loop  
2 iteration of loop  
3 iteration of loop
```

```
Goodbye
```

Pitajte stručnjaka

Pitanje: Da li mogu da koristim komandu **break** za zaustavljanje **foreach** petlje?

Odgovor: Komanda **break** je jedini način za zaustavljanje **foreach** petlje pre nego što se dođe do kraja liste. U *fnamesAM* skriptu koji sledi, korisnik želi da štampa samo imena fajlova koja počinju slovima iz prve polovine alfabeta: imena fajlova koja počinju sa slovima između a i m. Struktura **foreach** će štampati sva imena fajlova iz njene liste. Da biste je zaustavili, koristite komandu **break**.

```
#  
  
foreach filename ( * )  
    if ( $filename =~ [n-z]* ) break  
    echo $filename  
end  
  
echo "Goodbye"
```

Sledi izvršenje *fnamesAM* skripta:

```
> ls  
lib.c main.c resume termpaper  
> fnamesAM  
lib.c  
main.c  
Goodbye
```

Projekat 4-2: Indeksiranje u TCSH shellu

Program *myindext* je TCSH shell verzija BASH *myindex* programa opisanog u trećem poglavljiju. Izvršava iste funkcije, automatsko generisanje web stranice koja može da funkcioniše kao indeks za biranje ostalih web stranica. Pretpostavlja se da postoje poddirektorijumi koji sadrže web stranice. Mnoge shell operacije su iste kao i u BASH shell programi. Komande **grep**, **echo**, **basename**, **cut** i **sed** koje su korišćene u *myindex* su zadržane i u **myindext**, i to neizmenjene.

Petlje i **if** strukture, kao i dodeljivanje, se veoma razlikuju u programu ovog poglavљja. U *myindext* postoje dve ugnezđene **foreach** petlje, slično **for** petlji u BASH programu. Operacije dodeljivanja koriste komandu **set** sa znakom **=**. Uslovi za **if** se navode u malim zagradama, a standardni operatori slični onima u C-u rade sa njima. Zatim se postavlja ključna reč **then** u istoj liniji kao kod testa BASH shella, a **if** uslov se završava ključnom reči **endif**.

Korak po korak

1. Izaberite direktorijume u tekućem direktorijumu i proverite da li sadrže web stranice (svi fajlovi sa ekstenzijama **.html** ili **.htm**). Imena ovih fajlova se postavljaju u string koji sadrži HTML komande, **hrefs** za selektovanje i prikazivanje fajlova. HTML reference su organizovane u listu sa **** i **** oznakama, a svakom elementu prethodi oznaka ****. Imena direktorijuma se koriste kao zaglavje, oznaka **<h1>**.
2. Koristite komandu **foreach** sa ***** radi poklapanja sa svim fajlovima i nazivima direktorijuma, kako bi se svi uključili u promenljivu **\$i**.
3. Ako pronađete direktorijum, pretražite da li ima HTML fajlova u unutrašnjoj **foreach** petlji. Postavite HTML fajlove u promenljivu **j**.
4. Pretražite HTML fajlove prema naslovima i koristite link tekst u indeksu. Za pretraživanje pojave šablona **<TITLE>** koristite komandu **grep**. Ako ga pronađete, vratite ga u promenljivu **tline**. Ako ne nađete šablon, vratite 1 u statusnu promenljivu **\$?**
5. Proverite statusnu promenljivu **\$** kako biste videli da li je izvršenje komande **grep** bilo uspešno. Ako jeste, koristite skup **sed** komandi za izvlačenje prethodnih **<TITLE>** i krajnjih **</TITLE>** oznaka zajedno sa ostalim karakterima. Tekst naslova ostaje i dodeljuje se promenljivoj **ntitle**.
6. Ako komanda **grep** ne pronađe liniju **<TITLE>**, dodelite ime fajla promenljivoj **ntitle**. Najpre koristite komandu **basename** za izvlačenje imena fajla iz puta, tako što se uklanjaju imena direktorijuma koji prethode fajlu. Zatim koristite komandu **cut** za uklanjanje ekstenzije (štampa se prvo polje korišćenjem tačke kao graničnika). Uklanjuju se obe ekstenzije, i **.html** i **.htm**.
7. Posle dodeljivanja teksta promenljivoj **ntitle**, koristite tekst link u **href** HTML liniji. Ova linija počinje sa **** oznakom, ukazujući da se radi o listi elemenata.

Ovde je prikazan *myindex.t* skript:

```
#!/bin/tcsh

echo '<HTML>'
echo '<HEAD>'
echo '<TITLE>My Index</TITLE>'
echo '</HEAD>'
echo '<BODY>'
echo '<H1>Index of HTML Files</H1>

foreach i ( * )
if ( -d $i ) then
echo "<h2>$i</h2>"
echo '<ul>'
foreach j ( $i/*.htm* )
if ( -f $j ) then
set tline = `grep '<TITLE>' $j`
```

```
if ( $? == 1 ) then
    set ntitle = `basename $j | cut -f1 -d"."'
else
    set ntitle = `echo $tline | sed 's/^.*<TITLE>//' |
                  sed 's/<\/TITLE>.*$/'''
endif
echo "<li><a href=$j>$ntitle</a>"
endif
end
echo '</ul>'
endif
end

echo '</BODY>'
echo '</HTML>'
```

GLAVNA PROVERA

1. Koja se komanda koristi za čitanje ulaza u TCSH shell skriptu?
2. Možete li da pozivate više elemenata niza odjednom?
3. Kako može da se odredi broj argumenata koje korisnik unese u komandnoj liniji kada se izvršava skript?
4. Kako se definije promenljiva koja može da se poziva iz podshellova?
5. Koja je razlika između operatora =, == i -=?
6. Kako ćete da proverite da li je argument, kojeg je korisnik isporučio skriptu, naziv direktorijuma?

