Командни процедури во UNIX

## Користење аргументи

Командните процедури можат да примаат аргументи од командна линија. При пишување на вакви скрипти по правило се користи слична синтакса како кај останатите UNIX команди. За процесирање на аргументите на комадна линија постојат оператори и вградени системски променливи преку кои им се пристапува на аргументите. Променливата $# го содржи бројот на аргументи, а $\* самите аргументи. Секој аргумент на командната линија е и индивидуално достапен преку променливите $1, $2 итн. Променливата $0 го содржи името на самата команда.

## Користење опции

Вообичаената синтакса на UNIX командите покрај аргументите е да содржат и опции. Преку функцијата getopts (get options) може да се врши парсирање на командната линија и процесирање на наведените опции. Функцијата getopts има два аргументи. Првиот е листа на дозволените букви, а после оние за кои мора да има аргументи се става “:”.

На пример ако некоја скрипта има три можни опции така што првата (-y) и третата (-x) се без аргумент, а втората со аргумент (-z), тогаш првиот аргумент на функцијата getopts би бил "yz:x". Вториот аргумент е привремена променлива во која се сместува буквата која се обработува во циклусот. Циклусот while се извршува по еднаш за секоја опција на командната линија. Процедурата opcii.sh:

|  |
| --- |
| #!/bin/bash  while getopts xz:y c  do  case $c in  x) XFLAG=true  echo $XFLAG  ;;  y) YFLAG=true  echo $YFLAG  ;;  z) ZFLAG=true  echo "Argumentot na opcijata –z e : $OPTARG"  ;;  \*) USAGE="Povikajte ja skriptata so nekoja od opciite x, y ili z"  echo $USAGE  exit 2  esac  done |

може да се извршува на следните начини:

|  |
| --- |
| student@os:~$ ./opcii.sh  student@os:~$ ./opcii.sh -x  student@os:~$ ./opcii.sh -y  student@os:~$ ./opcii.sh –z test  student@os:~$ ./opcii.sh –xy  student@os:~$ ./opcii.sh –h |

Ако се зададе:

|  |
| --- |
| student@os:~$ ./opcii.sh -z |

ќе се иницира прикажување на вредноста на променливата USAGE во која функцијата getopts сместува порака за грешка при погрешна употреба на опциите. Променливата OPTARG го содржи аргументот соодветен на таа опција.

### Извршување команди

Освен со последователно сместување на повеќе команди во еден ред одвоени со “;”, секвенцијално извршување на команди може да се изведе и на други начини.  
  
За извршување на команда паралелно со друг процес се користи & како суфикс на командата:

*команда &*

Секвенцата *команда1 && команда2* прво ја извршува команда1. Ако таа заврши со излезен код 0, тогаш се извршува и команда2, во спротивно не се извршува команда2. Тоа е еквивалентно со:

if *команда1* then

*команда2*

fi

Типично се користи за кратенки од типот:

|  |
| --- |
| student@os:~$ test –f file.txt && rm file.txt |

со што се брише датотеката со име file.txt само ако постои.

Секвенцата *команда1 || команда2* ја извршува команда1. Ако не успее, ја извршува команда2.  
  
Пример:

|  |
| --- |
| student@os:~$ test –f file.txt || (echo can\'t find file.txt && exit 1) |

Со сместување на команда во загради (...) се предизвикува извршување на командата во нов процес. Тоа се користи во случаи кога е потребна измена на околината за некоја команда без да се влијае на други команди. На пример, за одредување на патеката на именикот во погорно ниво може да се употреби:

|  |
| --- |
| student@os:~$ (cd..; pwd) |

break излегува од циклус. Се користи за прекинување на циклус во точка во внатрешноста на циклусот.

continue го враќа извршувањето на почеток на циклусот.

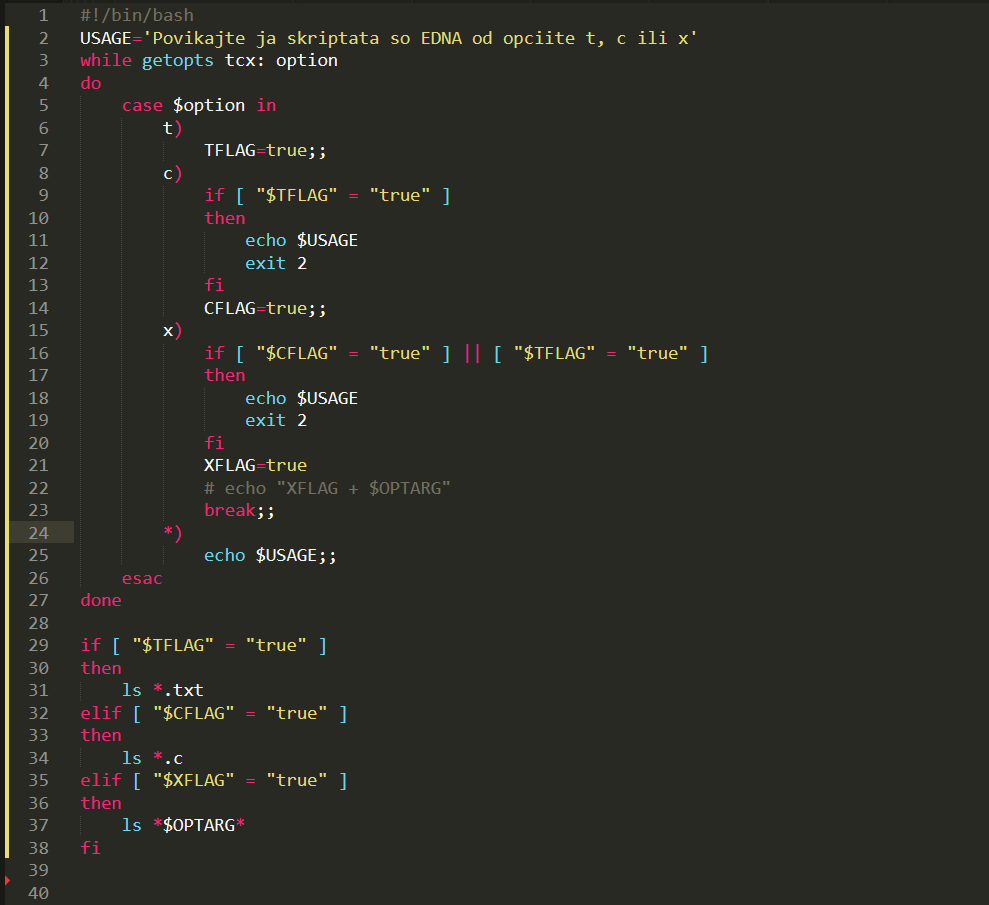
Пример:

|  |
| --- |
| #!/bin/bash  for file in \* do  if [ !-r $file ] then  echo can\'t find $file  continue  fi  done |

**Задачи**

1. Да се направи скрипта која ги печати само имињата на датотеките кои имаат привилегија за извршување. Каков тип на датотеки ќе се разгледуваат зависи од вредноста на зададените опции при повикувањето на скриптата
   * Доколку опцијата е -t – се разгледуваат само .txt датотеки
   * Доколку опцијата е -c – се разгледуваат само .c датотеки
   * Доколку опцијата е -x – се разгледуваат само датотеките чие име го содржи аргументот на опцијата

Доколку нема такви да се испечати соодветна порака.



1. Да се напише командна процедура (shell script), со користење на sed, која од даден именик (наведен како прв аргумент од командна линија) ќе ги ископира сите датотеки на кои корисникот ги има сите привилегии, во друг именик (вториот аргумент од командната линија). Доколку вториот именик кој е наведен не постои, потребно е истиот да се креира. Доколку не бидат наведени двата аргументи потребно е да се отпечати упатство за користење.

Напомена: Да се претпостави дека датотеките немаат whitespace карактери во имињата. Да се води сметка за постоењето на именикот од кој треба да се ископираат фајловите.

1. Да се напише Shell скрипта која ќе ги организира датотеките од тековниот именик според екстензијата. Организацијата се врши преку преместување на датотеките во соодветни именици, чии имиња се еднакви на екстензиите. Имениците треба да се преместат во именик чија локација е зададена како прв аргумент на скриптата. Доколку овој именик не постои да се креира. Доколку постои, да се избрише сите датотеки и именици кои се наоѓаат во него.

Пример: Именикот argument/mp3/ треба да ги содржи сите .mp3 датотеки, додека пак именикот argument/jpg/ треба да ги содржи сите .jpg датотеки.

1. Да се напише командна процедура во Unix што ќе ги запише броевите на Фибоначи, помали од бројот даден како прв аргумент на процедурата, во датотека дадена како втор аргумент на процедурата. Низата на Фибоначи започнува со две единици, а секој нареден член претставува збир од претходните два. Доколку датотеката не постои, потребно е да се креира, а доколку постои потребно е низата да се запише на крајот од истата.
2. Да се напише Shell скрипта која ќе креира бинарно стебло од именици со длабочина зададена како прв аргумент на скриптата. Секој директориум треба да има име кое започнува со вториот аргумент, по што следува редниот број на нивото во стеблото во кое е именикот и на крај редниот број на директориумот во соодветното ниво (види слика).

Пример: ./steblo.sh 3 ime

Треба да ја креира следната структура:

ime34

ime33

ime32

ime31

ime22

ime21

ime11