Odpowiedzi i podpowiedzi do zadań

Niniejszy dokument można wykorzystać w sytuacji, gdy będziemy potrzebować pewnego rodzaju wsparcia merytorycznego i np. porównania z samodzielnie opracowanym rozwiązaniem. Rozwiązania przedstawione tutaj są jedynie przykładowymi.

1 Wybrane zadania z zajęć

Zadanie 3

3. Napisz skrypt, który przyjmuje trzy argumenty:

• ścieżkę do pliku regularnego, który użytkownik może czytać,

• ścieżkę do katalogu, gdzie użytkownik ma prawo do zapisu,

• liczbę większą od zera.

Podany (argument 1.) plik kopiuje do podanego katalogu (argument 2.) tyle razy, ile wynosi argument 3. Na końcu nazwy każdej kopii dołącz sufiks -kopia- i numer kopii.

Przykładowe wywołanie:

./skrypt ala/ma/kota.txt /tmp/moj_katalog 5

powinno skutkować następującą zawartością katalogu /tmp/moj_katalog:

kota.txt-kopia-1

kota.txt-kopia-2

kota.txt-kopia-3

kota.txt-kopia-4

kota.txt-kopia-5

przygotowania:

```
echo "Testowa zawartosc pliku XD" > src.txt
mkdir subdir
```

wywołanie:

```
./zadanie_3.sh ~/c6/src.txt ~/c6/subdir 3
```

skrypt:

```
#!/bin/bash
```

```
if [[ $# != 3 ]] ; then
        echo "Błąd: skrypt oczekuje 3 argumentów"
        exit 1
fi

if [[ ! -r $1 ]]; then
        echo "brak uprawnień do odczytu pliku z arg 1"
        exit 2
fi

if [[ ! -w $2 ]]; then
        echo "brak uprawnień do zapisu katalogu z arg 2"
        exit 3
fi

if (( $3 < 0 )); then
        echo "niewłaściwa liczba kopiowań"
        exit 4
fi

filename=$( echo $1 | awk -F'/' '{print $NF}' )

echo "sama nazwa pliku: $filename"

i=1

while (($i <= $3))
do
        newname="$filename-kopia-$i"
        cp $1 "$2/$newname"
        i=$((i+1))
done

exit 0
```

Prosty test:

```
ada@computer:~/Desktop/unix2020_lab/c6_testing$ echo "Testowa zawartosc pliku XD" > src.txt
ada@computer:~/Desktop/unix2020_lab/c6_testing$ cat src.txt
Testowa zawartosc pliku XD
ada@computer:~/Desktop/unix2020_lab/c6_testing$ mkdir subdir
ada@computer:~/Desktop/unix2020_lab/c6_testing$ ls -l
total 228
-rw-------  1 ada ada    37 Feb 29  2020 input.txt
-rwx------  1 ada ada   762 Feb 29  2020 nakolos_a11_OLD.sh
-rwx------  1 ada ada   677 Feb 29  2020 nakolos_a11.sh
-rwx------  1 ada ada   707 Feb 29  2020 nakolos_a12.sh
-rwx------  1 ada ada   470 Feb 29  2020 nakolos_B11.sh
-rwx------  1 ada ada   209 Feb 29  2020 nakolos_B12.sh
-rwx------  1 ada ada   213 Feb 29  2020 spr3.sh
-rw-rw-r--  1 ada ada    27 Jan  1 20:05 src.txt
```

```
drwxrwxr-x  2 ada ada  4096 Jan  1 20:06 subdir
drwxrwxr-x 22 ada ada  4096 Jan  1 18:29 test5
drwxrwxr-x  2 ada ada  4096 Jan  1 18:29 testkat
drwxrwxr-x  2 ada ada  4096 Jan  1 18:29 testkat1
-rw-------  1 ada ada 71768 Feb 29  2020 unix_kartkowka_2_005.odt
-rwx------  1 ada ada   264 Feb 29  2020 zadanie1.sh
-rwx------  1 ada ada   526 Feb 29  2020 zadanie_3.sh
-rwx------  1 ada ada   700 Feb 29  2020 zadanie_6.sh
ada@computer:~/Desktop/unix2020_lab/c6_testing$ ./zadanie_3.sh src.txt subdir 3
sama nazwa pliku: src.txt
ada@computer:~/Desktop/unix2020_lab/c6_testing$ cd subdir/
ada@computer:~/Desktop/unix2020_lab/c6_testing/subdir$ ls -l
total 36
-rw-rw-r-- 1 ada ada 27 Jan  1 20:10 src.txt-kopia-1
-rw-rw-r-- 1 ada ada 27 Jan  1 20:10 src.txt-kopia-2
-rw-rw-r-- 1 ada ada 27 Jan  1 20:10 src.txt-kopia-3
ada@computer:~/Desktop/unix2020_lab/c6_testing/subdir$ cat src.txt-kopia-1
Testowa zawartosc pliku XD
ada@computer:~/Desktop/unix2020_lab/c6_testing/subdir$
```

Zadanie 4

4. Napisz skrypt, który w nieskończonej pętli będzie co 3 sekundy wypisze liczbę aktualnie obecnych procesów w systemie. Zakończenie działania skryptu powinno odbywać się poprzez kombinację klawiszy Ctrl-c.

Przypomnienie: żeby uśpić działanie skryptu na 3 s wydajemy komendę sleep 3

Zmodyfikujmy nieco fragment treści zadania, aby dodać element *user friendly*:

Zakończenie działania skryptu powinno odbywać się poprzez kombinację klawiszy Ctrl-c. ***Przy zakończeniu swojego działania skrypt powinien wypisywać komunikat o zakończeniu swojego działania. a wartością zwracaną przy takim zakończeniu powinno być 123.***

Przykładowa treść skryptu (uwaga, spoiler):

```
#!/bin/bash

trap 'echo "kończę" ; exit 123 ' SIGINT

#alternatywnie zamiast SIGINT można napisać 2
#trap 'echo "kończę" ; exit 123 ' 2

while [[ 1 ]]
```

```
do
#           echo "liczba procesów to: $(ps --no-headers -e | wc -l)"
  liczbaproc=$( ps --no-headers -e | wc -l )
  echo $liczbaproc
            sleep 3
done
```

Komentarz 1:

```
trap 'echo "kończę" ; exit 123 ' SIGINT
```

trap:

- arg 1: co robić jak pojawi się sygnał, tu 'echo "kończę" ; exit 123 '
- arg 2: określa nr sygnału lub mnemonik sygnału, tu: SIGINT

Zamiast SIGINT można napisać nr sygnału SIGINT, czyli tutaj byłoby to 2.

Komentarz 2:

pętla nieskończona – przykład realizacji (jeden z wielu): while [[1]]

Komentarz 3:

Korzystamy z zagnieżdżenia instrukcji: $(instr. wewn. wykonana najpierw)

Prosty test:

```
ada@computer:~/Desktop/unix2020_lab/c6_testing$ ./zadanie_4.sh
280
280
280
^Ckończę
ada@computer:~/Desktop/unix2020_lab/c6_testing$ echo $?
123
```

Zadanie 5

5. Stwórz skrypt, który jako argument wywołania otrzymuje numer UID. Jeśli został on uruchomiony bez argumentu, to pyta o UID użytkownika. Skrypt, korzystając z pliku /etc/passwd, wypisuje zawartość 5. kolumny linii opisującej użytkownika o podanym UID. Jeśli jako argument podane zostanie więcej numerów UID, wypisz analogiczną informację dla wszystkich wskazanych użytkowników.

Przykładowa treść skryptu (uwaga, spoiler):

Wersja bez zmiennej pomocniczej w awk:

```
#!/bin/bash
uids=""
if (( $# == 0 )); then
 echo -n "Podaj listę UID: "
 read uids
else
 uids=$*
fi
for i in $uids ; do
 echo $( cat /etc/passwd | awk -F':' "{ if( \$3 == $i) print \$5 }" )
done
exit 0
```

Komentarz 1: W **awk** mamy cudzysłów podwójny, aby bash mógł rozwinąć $i jako i-ty argument.

Komentarz 2: odczytujemy komendą **read**, a użytkownik ma podać listę liczb ze spacjami

Komentarz 3: aby uniknąć escapowania, można zastosować zmienną w awk, np.

```
echo $( cat /etc/passwd | awk -F':' -v "oneuid=$i" '{ if( $3 == oneuid) print $5 }' )
```

Zadanie 6

6. Utwórz skrypt, który wypisuje z katalogu, w którym został uruchomiony, nazwy wszystkich plików i katalogów z zaznaczeniem, czy jest to plik regularny, czy katalog. Sam skrypt powinien zostać na takiej liście pominięty. Dla plików wypisz dodatkowo prawa dostępu dla właściciela.

Przygotowane pliki (podświetlono kilka linii do porównania):

```
ada@computer:~/Desktop/unix2020_lab/c6_testing$ ls -l
total 212
-rw-------  1 ada ada    37 Feb 29  2020 input.txt
-rwx------  1 ada ada   762 Feb 29  2020 nakolos_a11_OLD.sh
-rwx------  1 ada ada   677 Feb 29  2020 nakolos_a11.sh
-rwx------  1 ada ada   707 Feb 29  2020 nakolos_a12.sh
-rwx------  1 ada ada   470 Feb 29  2020 nakolos_B11.sh
-rwx------  1 ada ada   209 Feb 29  2020 nakolos_B12.sh
-rwx------  1 ada ada   213 Feb 29  2020 spr3.sh
-r--------  1 ada ada    27 Feb 29  2020 src.txt
drwxrwxr-x 22 ada ada  4096 Jan  1 18:29 test5
drwxrwxr-x  2 ada ada  4096 Jan  1 18:29 testkat
drwxrwxr-x  2 ada ada  4096 Jan  1 18:29 testkat1
-rw-------  1 ada ada 71768 Feb 29  2020 unix_kartkowka_2_005.odt
-rwx------  1 ada ada   264 Feb 29  2020 zadanie1.sh
-rwx------  1 ada ada   700 Feb 29  2020 zadanie_6.sh
```

Przykładowy output („UAP" to wymyślony roboczy akronim dla określenia *user access permision*):

```
ada@computer:~/Desktop/unix2020_lab/c6_testing$ ./zadanie_6.sh
script command: ./zadanie_6.sh
script name only: zadanie_6.sh
plik regularny input.txt UAP: READ WRITE
plik regularny nakolos_a11_OLD.sh UAP: READ WRITE EXECUTE
plik regularny nakolos_a11.sh UAP: READ WRITE EXECUTE
plik regularny nakolos_a12.sh UAP: READ WRITE EXECUTE
plik regularny nakolos_B11.sh UAP: READ WRITE EXECUTE
plik regularny nakolos_B12.sh UAP: READ WRITE EXECUTE
plik regularny spr3.sh UAP: READ WRITE EXECUTE
plik regularny src.txt UAP: READ
katalog test5
katalog testkat
katalog testkat1
plik regularny unix_kartkowka_2_005.odt UAP: READ WRITE
plik regularny zadanie1.sh UAP: READ WRITE EXECUTE
```

Hints:

- filtrowanie samej nazwy:
  `awk -F'/' '{print $NF}'`
- listowanie plików:
  `for i in * ; do`
- listowanie plików ale nie aktualnego skryptu:
  `if [[ (-f $i) && ($i != $onlyname) ]] ; then`
- listowanie katalogów:
  `if [[ -d $i ]] ; then`
- jak sprawdzić uprawnienia? np. grepem:
  - **^.r.*** - właściciel ma prawo read
  - **^..w.*** - właściciel ma prawo write
  - **^...x.*** - właściciel ma prawo execute

Teraz warto spróbować napisać ten skrypt samodzielnie, a następnie porównać z przykładową odpowiedzią poniżej.

```
#!/bin/bash
```

```
line=""
echo "script command: $0"
onlyname=$(echo $0 | awk -F'/' '{print $NF}')
echo "script name only: $onlyname"

for i in * ; do
	if [[ (-f $i) && ($i != $onlyname) ]] ; then
		echo -n "plik regularny $i"
		line=$(ls -l $i)
		#echo " line: $line"
		echo -n " UAP: "
		apread=$(echo $( echo $line | grep -E '^.r.*') )
		if [[ -n $apread ]] ; then
			echo -n "READ "
		fi
		apwrite=$(echo $( echo $line | grep -E '^..w.*') )
		if [[ -n $apwrite ]] ; then
			echo -n "WRITE "
		fi
		apexe=$(echo $( echo $line | grep -E '^...x.*') )
		if [[ -n $apexe ]] ; then
			echo -n "EXECUTE "
		fi
		echo ""

	fi
	if [[ -d $i ]] ; then
		echo "katalog $i"
	fi
done
exit 0
```

Zadanie 7

7. Napisz skrypt, który wyświetli prawa dostępu do wskazanego jako argument pliku z punktu widzenia użytkownika, który wywołał skrypt. Jeśli nie został podany żaden argument, skrypt powinien o niego poprosić.

Tutaj warto odróżniać: ***prawa dostępu do wskazanego jako***

argument pliku z punktu widzenia użytkownika, który wywołał skrypt (czyli to, co teraz mamy w treści zadania)

vs

to, co było w zadaniu 6: ***prawa dostępu dla właściciela***

```
#!/bin/bash
```

```
thefile=""

if (( $# == 0 )) ; then
   echo "Nie podano argumentu, podaj nazwe pliku: "
   read thefile
elif (( $# == 1 )) ; then
   thefile=$1
else
  echo "Nieprawidlowa liczba argumentow."
  exit 1
fi


if [[ ! -e $thefile ]] ; then
   echo "wskazany plik $thefile nie istnieje"
   exit 2
fi

echo -n "plik: $thefile, moje uprawnienia do tego pliku (uzytkownik $USER): "

if [[ -r $thefile ]] ; then
  echo -n "READ "
fi

if [[ -w $thefile ]] ; then
  echo -n "WRITE "
fi

if [[ -x $thefile ]] ; then
  echo -n "EXECUTE "
fi


echo ""

exit 0
```

Prosty test:

```
ada@computer:~/Desktop/unix2020_lab/c6_testing$ ./zadanie_7.sh
Nie podano argumentu, podaj nazwe pliku:
src.txt
plik: src.txt, moje uprawnienia do tego pliku (uzytkownik ada): READ WRITE
ada@computer:~/Desktop/unix2020_lab/c6_testing$ ./zadanie_7.sh src.txt
plik: src.txt, moje uprawnienia do tego pliku (uzytkownik ada): READ WRITE
```

Zadanie 8

```
#!/bin/bash

#tested with: ./zadanie_8.sh "input.txt" 7 "testkat"
```

```
echo "plik do skopiowania: $1"
echo "długosc czesci w bajtach: $2"
echo "katalog docelowy: $3"

FSIZE=$(du -sb input.txt | cut -f1)

echo "file size: $FSIZE"

RUNS=$(( FSIZE / $2 + 1 ))
echo "RUNS: $RUNS"

PASS=1

for (( PASS=1; $PASS<=RUNS; PASS=$PASS+1 )) ; do
  echo "PASS=$PASS"
  dd conv=noerror if=$1 of=$3/$1.$PASS bs=$2 skip=$((PASS - 1)) count=1 2>/dev/null
done
```

Fragment w wersji alternatywnej, z while:

```
while (( PASS <= RUNS )) ; do
  echo "PASS=$PASS"
  dd conv=noerror if=$1 of=$3/$1.$PASS bs=$2 skip=$((PASS - 1)) count=1 2>/dev/null
  PASS=$((PASS + 1))
done

exit 0
```

2 Zadania sprawdzające

Zadanie sprawdzające 1

1. Napisz skrypt, który co 2 sekundy wypisuje listę zalogowanych użytkowników razem z liczbą procesów, które zostały przez nich uruchomione, a w przypadku wciśnięcia przez użytkownika Ctrl-c wypisze aktualną datę i zakończy działanie.

```
#!/bin/bash
trap 'date; exit 0' SIGINT
while [[ 1 ]] ; do
	userlist=$(who | awk '{print $1}' | tr '\n' ' ')
	#echo "logged user list: $userlist"
	for i in $userlist ; do
		echo -n  "user $i : "
		echo $(ps --no-headers -u $i -o user,pid,state | wc -l)
	done
	sleep 2
done
```

Proste testy

Warto najpierw sprawdzić jak działa samo polecenie tworzące listę zalogowanych użytkowników, np.

```
userlist=$(who | awk '{print $1}' | tr '\n' ' ')
```

Teraz lista użytkowników jest w zmiennej userlist i możemy ją wypisać w konsoli, np.

```
echo "logged user list: $userlist"
```

W skrypcie jej wypisywanie zostało „zakomentowane", ponieważ w treści zadania nie było mowy o jej wypisywaniu – powinniśmy wypisywać tylko te informacje, które są wymagane w treści zadania.

Następnie, dla każdego użytkownika, w pętli for wypisujemy wymagane informacje. W poleceniu ps możemy wskazać, dla którego użytkownika chcemy wypisać informacje, używając np. opcji **-u** i podając nazwę użytkownika – to może znacznie uprościć skrypt.

Zadanie sprawdzające 2

Napisz skrypt, który otrzyma ścieżki do katalogów i dla każdego z nich policzy, ile jest w nich (oraz rekursywnie w podkatalogach) plików regularnych,

Poniżej przedstawiono sam rdzeń rozwiązania
Uwaga: rozwiązanie nie zawiera zabezpieczeń przed podaniem nieprawidłowych argumentów – na sprawdzianie należy takie zabezpieczenia uwzględnić, jeśli będzie wymagała tego treść zadania.

```
#!/bin/bash

#...sprawdzenie poprawnosci argumentow powinno byc np. tutaj...

echo "lista arg: $*"
for i in $* ; do
      echo -n "W katalogu $i jest "
      echo "$(find $i -type f | wc -l) plików regularnych"
done
```

Zadanie sprawdzające 3

Napisz skrypt, który skopiuje plik, podany jako 1. argument, do wskazanego katalogu docelowego, podanego jako 2. argument (skrypt powinien obsługiwać sytuacje wyjątkowe związane z brakiem dostępu do wskazanego pliku i katalogu i wyświetlać wtedy stosowne komunikaty).

(Do samodzielnego rozwiązania)