Linux
Polecenia

Opracował: Andrzej Nowak

Operatory zmiany sposobu przypisania standardowych strumieni >,<,>>

Przykłady:

2> plik przypisuje standardowe wyjście błędów do pliku,

1>&2 przypisanie standardowe wyjście do tego, do czego jest przypisywany standard. błąd,

2>&1 przypisanie stand. błędu do tego, do czego jest przypisane standardowe wyjście.

Tworzymy katalog zaj_5 i przechodzimy do niego:

```
$ mkdir zaj_5 && cd zaj_5
```

Jeżeli pierwsze polecenie powiodło się to wykona drugie

```
$ cp
$ cp 2> errplik
$ cat errplik
$ cp 2> /dev/null    (urządzenie puste)
$ find / -name \*.c -print
$ find / -name \*.c -print 2> /dev/null/ (błąd)
$ find / -name \*.c -print > zanal 2> /dev/null/ (błąd)
$ cat znal
$ rm znal
$ find / -name \*.c -print > znal 2>&1 (błąd tam dokąd wyjście)
$ echo ”Plik wymaga argumentu” 1>&2 (wyślij standardowe wyjście do standardowego błędu)
```

Polecenie exec

exec- wykonuje dane polecenie zamieniając bieżący program (proces) (zamiast tworzenia nowego procesu); to polecenie jest użyteczne również dla otwierania, zamykania i kopiowania deskryptorów plików.

Przejdźmy do katalogu domowego.

```
$ ls

$ exec ls

$ tcsh

$ bash

$ ps

$ exec ls

$ ps

$ exit

$ exec < plika   (otwórz plika dla czytania ze stand. wejścia)

$ exec > plikx       (otwórz plikx dla pisania na stand. wyjście)

$ exec 3< plik       (otwórz plik poprzez fd3 w celu czytania w
charakterze stand. wejścia)

$ exec 4> nowy_plik (otwórz nowy_plik poprzez fd4 w celu pisania
(stand. wyjście))

$ ls >&4        (wyjście ls jest skierowane do nowy_plik)

$ exec 5<&4       (fd5 będzie kopią fd4)

$ exec 3<&-       (zamknij fd3)

$ exec > /dev/tty

$ ls

$ tcsh

$ bash
```

Utwórzmy plik proc_5:

```
pwd
echo czesc
date
```

```
$ exec proc_5
```

Otwiera standardowe wejście dla proc_5; czyta z pliku zamiast z klawiatury; polecenia z pliku są wykonywane w miejsce bieżącego shell-a; wyjście z ostatniego polecenia powoduje wyjście z shell-a

```
# exec 3> plikx
```

fd3 jest przypisane plikowi **plikx** i zostaje on otwarty w charakterze standardowego wyjścia

```
$ ls
$ who >&3
$ cat plikx

$ date >&3
$ cat plikx

$ exec 3>&-
```

fd3 jest zamknięty

```
$ date

$ exec 3< plikx
```

fd3 jest otwarte w celu czytania; wejście z pliku **plikx**

```
$ cat <&3

$ exec 3<&-
```

fd3 jest zamknięty; jednocześnie **plikx** będzie zamknięty

```
$ date >f3
```

Dokument ”odtąd – dotąd”(ang. Here document)

Przejdź do katalogu zaj_5

```
$ cat << KONIEC
> Czesc $LOGNAME
> Jest godzina `date`
> Nie moge dluzej czekac!!!
> KONIEC
```

Polecenie read

Napisz skrypt super

```
#!/bin/bash
# nazwa skryptu: super
echo "Czy jestes szczesliwy?"
read odpowiedz
echo "$odpowiedz jest sluszna odpowiedzia"
echo "Podaj swoje imie i nazwisko: "
read imie nazwisko
echo "Halo $imie"
```

```
$ ./happy
```

Arytmetyka stałoprzecinkowa: polecenia expr

```
$ expr 1 + 4
$ expr 1+4
$ expr 5 + 9 / 3
$ expr 5*4
$ expr 5 * 4
$ expr 5 \* 4 - 2
$ expr 11 % 3

$ num=1
$ num=`expr $num + 1`
```

```
$ echo $num
```

Inne przykłady użycia polecenia expr:

- Pierwsze działanie będzie dzieleniem:

```
$ expr 5 + 10 / 2
```
wynikiem jest 10

- Pierwsze będzie dodawanie:

```
$ expr \( 5 + 10 \) / 2
```
wynikiem jest 7

- Dodaj 1 do zmiennej i; Przykład użycia expr w skryptach:

```
$ i=`expr $i + 1`
```

- Wyświetli 1 (prawda) jeżeli zmienna a będzie miała wartość "hello":

```
$ expr $a = hello
```

- Wyświetli 1 (prawda) jeżeli suma zmiennej b i liczby 5 będzie większa lub równa 10:

```
$ expr $b + 5 \>= 10
```

- W poniższym przykładzie, zmienna p jest ciągiem "version.100". To polecenie wyświetli liczbę znaków w zmiennej p:

```
$ expr $p : '.*'
```
wynikiem jest 11

- Dopasuje znaki i wyświetli pasujące:

```
$ expr $p : '\(.*\)'
```
wynikiem jest "version.100"

- Wyświetli liczbę małych liter w zmiennej p:

```
$ expr $p : '[a-z]*'
```
wynikiem jest 7

- Dopasuje małe litery w zmiennej p:

```
$ expr $p : '\([a-z]*\)'
```
wynikiem jest "version"

- Utnie zawartość zmiennej x jeżeli zawiera pięć lub więcej znaków; jeżeli nie, wyświetli zmienną x.

```
$ expr $x : '\(.....\)' \ | $x
```

- W skrypcie, skróci nazwę plikom do początkowych pięciu znaków:

```
$ mv $x `expr $x : '\(.....\)' \ | $x`
```

Ćwiczenia

Ćwiczenie 1

Utwórz skrypt o nazwie **policz** pobierający dwie wartości wprowadzone jako parametry skryptu, a następnie wypisujący w kolejnych wierszach sumę, różnicę, iloczyn oraz iloraz obu składników.

Ćwiczenie 2

Napisz skrypt **rownanie**, który obliczy pierwiastek równania `Ax + B = 0`, jeżeli on istnieje.

Przeanalizuj parametry A i B równania i wszystkie możliwe sytuacje związane z ich

wartościami, a w szczególności:

- jeżeli uruchomiono skrypt bez podania wartości A i B, podaj użytkownikowi sposób użycia

  skryptu,

- jeżeli podano wartość tylko jednego parametru to przyjmij tę wartość jako parametr A, a

  B=0

- wynik podaj w postaci liczba1/liczba2 np. 3/2.