

Wstęp do systemu operacyjnego UNIX

Laboratorium 8:

Programowanie w językach interpreterów poleceń

Interpretery poleceń są wyposażone w interpretery prostych języków programowania. Składnia języka programowania stała się nawet podstawą systematyki interpreterów. Według niej mamy zatem interpretery rodziny `C` (`csh`, `tcsh`), których język programowania przypomina język programowania `C` i pozostałe należące do rodziny `sh` (`sh`, `ksh`, `bsh`, `bash`).

Niezależnie od interpretera, skrypt (jego program) zapisujemy w pliku tekstowym, pamiętając aby na końcu ostatniej linii dać znak przejścia do nowej linii.

Istnieje kilka sposobów uruchamiania programów dla interpretera poleceń.

1. Przez podanie nazwy skryptu w linii komend jako komendy do wykonania. Przed uruchomieniem skryptu musimy zmienić atrybuty pliku tak, aby ten kto go uruchamia miał prawo wykonywania i odczytu. Do interpretacji zostanie wywołany interpreter taki jak podstawowy (logujący) użytkownika uruchamiającego skrypt.
2. Jak w poprzednim przypadku, ale jeśli chcemy aby skrypt był interpretowany przez wskazany interpreter to pierwsza linia skryptu od pierwszej kolumny musi zawierać specyfikację bezwzględnej ścieżki dostępu do tego interpretera jak poniżej:

```
1  #!/bin/bash
2  # ten skrypt interpretuje zawsze bash
3
```

3. W linii komend uruchamiamy interpreter poleceń podając jako argument jego wywołania ścieżkę dostępu do uruchamianego skryptu. Wówczas do zinterpretowania skryptu zostanie oczywiście uruchomiony wskazany interpreter poleceń niezależnie od tego, czy pierwsza linia skryptu zawiera specyfikację interpretera poleceń czy nie. W tym przypadku plik nie musi mieć prawa wykonywania. Ten sposób uruchamiania jest wykorzystywany podczas debuggowania skryptu. Interpreter uruchamiamy wówczas z opcjami `-vx`, co daje następujący, przykładowy wynik:

```
1  [bory@thorin skrypty]$ bash -vx skrypt1 aa
2  #!/bin/tcsh
3  # wypisanie argumentow dla bash
4  licznik=1
5  + licznik=1
6  while [ $1 ]
7  do
8      echo "Argument ${licznik} to $1"
9      licznik='expr ${licznik} + 1'
10     shift
11 done
12 + '[' aa ']'
13 + echo 'Argument 1 to aa'
14 Argument 1 to aa
15 expr ${licznik} + 1
16 ++ expr 1 + 1
17 + licznik=2
18 + shift
19 + '[' ']'
20
```

1. Napisz dla dowolnego interpretera poleceń skrypty działające w sposób następujący:

1. Skrypt wypisuje liczbę otrzymanych argumentów. Jeśli został wywołany bez argumentów to wypisuje stosowny komunikat. Skrypt w kolejnych liniach wypisuje wszystkie argumenty z którymi został wywołany poprzedzając każdy z nich napisem `argument nr. x` – gdzie `x` jest numerem argumentu.
2. W pętli nieskończonej, co 30 sekund skrypt wypisuje liczbę procesów uruchomionych w systemie. Zakończenie działania - kombinacja klawiszy `Control-c`.
3. Skrypt jako argument wywołania otrzymuje `UID`. Jeśli został uruchomiony bez argumentu, to doczytuje go z klawiatury. Skrypt korzystając z pliku `/etc/passwd` wypisuje zawartość 5 kolumny linii opisującej użytkownika o podanym `UID`.
4. Jak w punkcie poprzednim, ale należy uwzględnić przypadek, w którym w linii komend podano kilka argumentów – `UID` kilku użytkowników.
5. Skrypt wypisuje z katalogu w którym został uruchomiony nazwy wszystkich plików i katalogów z zaznaczeniem, czy jest to plik czy katalog. Dla plików regularnych podaje prawa użytkownika (`rwX`) – użyj polecenia `test`.
6. * Napisz skrypt dzielący plik na kawałki (zawartość dużego pliku ma zostać zapisana do kilku plików o rozmiarach odpowiednio mniejszych). Skrypt posiada dwa argumenty wywołania. Pierwszy to nazwa pliku, a drugi to liczba kawałków. Do kopiowania użyj komendy `dd`:

```
1 dd conv=noerror if=$1 of=$1.$PASS bs=$CHUNK skip=$((PASS - 1)) count=1
2 2>/dev/null
```

gdzie: `PASS` oznacza numer kawałka pliku, `CHUNK` rozmiar kawałka pliku, a `conv=noerr` spowoduje, że komenda `dd` będzie kopiować tylko do końca pliku źródłowego. Jest to konieczne w przypadku, gdy ostatni kawałek jest mniejszy od zaproponowanego rozmiaru bloku (sytuacja często spotykana).

7. Dla interpretera poleceń `bash` napisz skrypt który w pętli nieskończonej co sekundę wypisuje napis ciągły. W przypadku otrzymania sygnału `USR1` lub `USR2` wypisuje napis `dostałem sygnał USRx` – gdzie `x` jest numerem otrzymanego sygnału (wskazówka: instrukcja `trap`).