

Wstęp do systemu operacyjnego UNIX

Laboratorium 5:

Programowanie w językach interpreterów poleceń

Interpreterzy poleceń są wyposażone w interpretery 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. 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 debugowania skryptu. Interpreter uruchamiamy wówczas z opcjami `-vx`, co daje następujący, przykładowy wynik:

```
1 [bory@thorin skrypty]$ bash -vx skrypt1 aa
2 #!/bin/bash
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 + '[', ']'

```

Napisz dla dowolnego interpretera polecenia skryptu działające w następujący sposób:

1. Skrypt wypisuje liczbę argumentów, z którymi został uruchomiony. Jeśli został uruchomiony bez argumentów to wypisuje stosowny komunikat. Skrypt w kolejnych liniach wypisuje wszystkie argumenty z którymi został uruchomiony poprzedzając każdy z nich napisem **argument nr. x** – gdzie x jest numerem argumentu.
2. W pętli nieskończonej, co 3 sekundy 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 lub należy przeczytać 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 regularny czy katalog. Dla plików podaje prawa dostępu użytkownika (rwx), który uruchomił skrypt.
6. * Skrypt ma podzielić plik na kawałki (zawartość dużego pliku ma zostać zapisana do plików o rozmiarach odpowiednio mniejszych). Skrypt musi zostać uruchomiony z dwoma argumentami. Pierwszy to nazwa pliku, który ma zostać podzielony, a drugi to rozmiar pojedynczego kawałka (mniejszego pliku) w bajtach. Do kopiowania najprościej użyć polecenia **dd** w postaci:

```

1 dd conv=noerror if=$1 of=$1.$PASS bs=$CHUNK skip=$((PASS - 1)) count=1 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** należy napisać skrypt który w pętli nieskończonej co sekundę wypisuje napis **ciagle zyje**. W przypadku otrzymania sygnału **USR1** lub **USR2** wypisuje napis **dostalem sygnal USRx** – gdzie x jest numerem otrzymanego sygnału (wskazówka: instrukcja **trap**).