Powłoki

W systemach UNIX podstawowym interfejsem pomiędzy systemem a użytkownikiem jest powłoka (ang. shell). Powłoki czytają komendy użytkownika (poprzez klawiaturę lub mysz) i mówią systemowi czego użytkownik oczekuje. W Linuksie dostępnych jest wiele powłok. Najpopularniejszą (instalowaną domyślnie) jest BASH ("Bourne Again SHell"). Większość komputerów działających pod Linuksem w sposób automatyczny konfiguruje i automatycznie uruchamia BASH.

Zestawienie powłok

sh - Powłoka Bourne'a. Jest to oryginalny shell, który został stworzony przez Steven`a R. Bourne'a około 1975 roku i działa w większości systemów UNIX. Powłoka ta stała się podstawowym interpreterem poleceń. Na początku był to jedyny wybór. Nie oferuje ona zbyt wielu działań takich choćby jak edycja wiersza poleceń. Ma dość duże możliwości, jeśli chodzi o obsługę skryptów. Shell Bourne'a wspomaga programowanie strukturalne udostępniając zmienne lokalne oraz globalne, które muszą być eksportowane. Shell Bourne'a implementuje konstrukcje sterujące **IF-THEN-ELSE, CASE, FOR, WHILE, UNTIL**. Wyrażenia warunkowe oblicza korzystając z poleceń **test** i **expr** systemu UNIX, czym różni się od shelli C i Korna, które te wyrażenia obliczają bezpośrednio.

csh - Powłoka C. Opracowana została na Uniwersytecie Kalifornijskim w Berkley przez Billy Joy`a. Powłoka ta posiada historię poleceń. Pamięta listę ostatnio wykonywanych poleceń i umożliwia cofanie się na tej liście oraz ponowne wykonywanie poleceń bez konieczności wpisywania ich z klawiatury. Można też wyświetlić tekst polecenia z listy, zmodyfikować go, a następnie wykonać zmodyfikowane polecenie. Shell C implementuje też mechanizm aliasów. Umożliwia kontrolę nad programami wykonywanymi w tle (odłączanymi od terminala) i w planie pierwszym (angażującymi terminal). W shellu C można dowolnie przenosić programy z pierwszego planu do tła czego nie można było robić w powłoce Bourne'a. Shell C implementuje dwa rodzaje zmiennych lokalne i globalne. Zmienne te są ustawiane za pomocą poleceń **set** i **setenv**. Polecenia powłoki C przypominają język programowania C. Powłoka zawiera wszystkie operatory warunkowe języka **C** (np.: **==, >**). Shell C implementuje konstrukcje sterujące **IF-THEN-ELSE, SWITCH, FOREACH, REPEATE i WHILE**. Wyrażenia warunkowe wewnątrz tych instrukcji są obliczane w sposób bezpośredni.

ksh - Powłoka Korna. Nazwa powłoki pochodzi od jej autora Davida Korna pracującego w Laboratoriach Bella. Jest rozszerzeniem powłoki Bourne'a. Umożliwia sterowanie wykonywaniem procesów, które można zatrzymywać, uruchamiać w tle, przywoływać i przerywać. Shell Korna zachowuje pełną funkcjonalność shella Bourne'a dołączając wiele udogodnień shella C. Jednocześnie zawiera wiele zupełnie nowych poleceń. W większości przypadków shell Korna jest szybszy niż shell C, ale wolniejszy niż shell Bourne'a. Shell Korna implementuje mechanizm edycji wiersza polecenia, który umożliwia modyfikowanie wprowadzanych poleceń za pomocą poleceń typowych dla kilku popularnych edytorów systemu UNIX. Udoskonalona jest historia poleceń. Możliwe jest bezpośrednie przechodzenie do dowolnych poleceń zapamiętanych na tej liście. Wyrażenia warunkowe obliczane są bezpośrednio.

rsh - Ograniczona powłoka Bourne'a. Jest specjalną odmianą powłoki Bourne'a. Stosowana do tworzenia środowiska użytkowego o ograniczonych możliwościach i prawach dostępu. Wykorzystywana przez użytkowników o niewielkich umiejętnościach np. wykonujących podstawowe prace biurowe.

tcsh. Napisana przez Williama Joya jest rozszerzeniem powłoki C. Zawiera 53 wbudowane polecenia (można sprawdzić wywołując polecenie builtins) i 18 opcji wywołania. Emuluje właściwości powłoki csh, ale zawiera znacznie więcej możliwości, w tym edytor linii poleceń z kontrolą błędów wpisywania.

bash - Bourne Again Shell. Słowo bash jest skrótem od Bourne Again SHell. Mówi ono o autorze pierwowzoru (Steve Bourne). Jest domyślnie stosowaną powłoką Linuxa. Zawiera 48 wbudowanych poleceń oraz 12 opcji wywołania. Za pomocą klawiszy kursora można przechodzić między poleceniami (dzięki wbudowanej historii poleceń). Można edytować polecenia. Jeśli zapomnimy nazwy programu można poprosić o jej automatyczne dokończenie przy pomocy klawisza TAB. Bash ma wbudowaną pomoc i listę znanych poleceń. Posiada niektóre zalety csh i ksh oraz zawiera wiele rozszerzeń oraz nowych rozwiązań. Jest to interpreter potężny i elastyczny, jeśli chodzi o programowanie, jak i o prace interaktywną. Wszystkie wbudowane polecenia powłoki Bourne'a są dostępne także w Bashu.

zsh. Jest to najnowsza powłoka napisana oryginalnie przez Paula Falstada. Zawiera 84 wbudowane polecenia. Można ją wywoływać z ponad 50 różnymi opcjami. Emuluje działanie sh i ksh. Posiada historię, edycję oraz kontrolę pisowni poleceń. Umożliwia też sterowanie wykonaniem procesów. Pozwala na zaawansowane przeszukiwanie plików.

Uruchamianie powłoki bash

Gdy wywołujemy powłokę, sprawdza ona zmienne systemowe i plik **/etc/passwd**, aby ustalić, kim jest użytkownik i gdzie powinna rozpocząć się jego sesja logowania. Następnie szuka w systemie domyślnych plików konfiguracyjnych (katalog /etc), po czym sprawdza pliki konfiguracyjne w katalogu macierzystym użytkownika. Pliki w katalogu **/etc** to podstawowa konfiguracja shella, która jest czytana przez bash przy każdorazowym logowaniu się użytkownika.

Pliki konfiguracyjne z katalogu /etc/:
profile - ten plik zawiera podstawowe ustawienia nadawane przez administratora.
bashrc - w pliku tym najczęściej znajduje się tylko definicja znaku gotowości. Mimo iż znajduje się ona także
w pliku profile, bash czasami nie wyświetla znaku gotowości, dlatego konieczne jest ustawienie go także w bashrc.
inputrc - zawiera opcje do edytora poleceń basha.

Pliki konfiguracyjne z katalogu użytkownika:
.bash_profile - wykonywany podczas otwierania sesji
.bash_logout - wykonywany podczas zamykania sesji
.bashrc - lepiej stosować .bash_profile

Opcje wywołania powłoki bash:
-norc Nie odczytuje plików inicjacyjnych ~/.bashrc w przypadku wywoływania interaktywnej powłoki
-rcfile NAZWA_PLIKU Wykonuje polecenia ze wskazanego pliku, zamiast z pliku ~/.bashrc
-noprofile Nie wczytuje systemowego pliku startowego /etc/profile ani żadnego osobistego pliku inicjacyjnego (~/.bash_profile,
~/.bash_login, ~/.profile)
-version Wyświetla informacje o wersji
-login Sprawia, że powłoka działa tak, jakby została wywołana bezpośrednio przez program login
-nobraceexpansion Nie rozwija elementów otoczonych nawiasami klamrowymi
-nolineediting Nie używa biblioteki GNU Readline do interaktywnego wczytywania linii poleceń
-c ŁAŃCUCH Po przetworzeniu innych opcji odczytuje i wykonuje polecenia zawarte w **ŁAŃCUCHU**, a następnie kończy działanie
-i Wymusza interaktywne działanie powłoki
-r Powłoka staje się powłoką okrojoną (restricted).
-s Jeśli występuje ta opcja lub po przetworzeniu opcji nie pozostają żadne argumenty, to polecenia odczytywane są ze
standardowego wejścia. Opcja ta umożliwia ustawienie parametrów pozycyjnych podczas wywołania powłoki interaktywnej.
-D Na standardowym wyjściu drukowana jest lista wszystkich podwójnie cytowanych łańcuchów poprzedzonych znakiem $.

Jeśli po przetworzeniu opcji pozostają jakieś argumenty, a nie podano ani opcji **-c** ani **-s**, to zakłada się, że pierwszy argument jest nazwą pliku zawierającego polecenia powłoki. Jeżeli bash został wywołany w taki sposób, to **$0** przypisywana jest nazwa pliku z poleceniami, a parametrom pozycyjnym pozostałe argumenty. Bash odczytuje i wykonuje polecenia z tego pliku, a następnie kończy pracę. Kod zakończenia basha jest wówczas kodem zakończenia ostatniego wykonanego polecenia skryptu. Jeśli nie wykonano żadnego polecenia, to kod zakończenia wynosi **0**.

Gdy bash wywoływany jest jako powłoka zgłoszeniowa lub jako powłoka nie-interaktywna z opcją **--login**, w pierwszej kolejności czyta i wykonuje polecenia z pliku **/etc/profile**, jeśli takowy istnieje. Następnie **~/.bash_profile, ~/.bash_login i ~/.profile**, w tej kolejności, po czym odczytuje i wykonuje polecenia z pierwszego istniejącego i dającego się odczytać. Podczas kończenia pracy powłoki zgłoszeniowej, bash czyta i wykonuje polecenia z pliku **~/.bash_logout**, jeśli taki istnieje.

Funkcjonalność

Dokańczanie poleceń

Dokańczanie poleceń jest bardzo wygodną i przyspieszającą pracę opcją powłoki Bash. Dzięki niej nie musimy już wpisywać pełnej nazwy programu lub ścieżki do katalogu, pliku. Wystarczy, że wpiszemy jego pierwszą literkę (lub literki) i wciśniemy klawisz **TAB**. Jeżeli można jednoznacznie określić nazwę zostanie to zrobione natomiast jeśli więcej programów zaczyna się tak jak to wpisaliśmy to powłoka wylistuje nam ich nazwy i poinformuje nas za

pomocą brzęczyka, że należy podać dodatkowe informacje. Jeśli wpiszemy nazwę która nie ma odniesienia w rzeczywistości i naciśniemy **TAB** interpreter poinformuje nas o błędzie wydając z brzęczyka systemowego charakterystyczny dźwięk. Jest to dla nas sygnał, że coś jest nie w porządku i należy poprawić.

Historia poleceń

Bash zapamiętuje wydane przez użytkownika polecenia. Dzięki temu za pomocą klawiszy kursora (góra - dół), możemy szybko przywołać polecenie, które już wcześniej wydaliśmy. Jest to szczególnie przydatne gdy w bardzo skomplikowanym poleceniu zmieniamy np. tylko nazwę pliku, na którym ma zostać wykonane, a cała składnia polecenia nie ulega modyfikacjom. Lista wydanych przez nas poleceń jest ładowana z pliku historii, który domyślnie nosi nazwę **.bash_history**. Parametry nazwy pliku historii oraz ilości przechowywanych w nim ostatnio wydanych poleceń ustalają zmienne **HISTFILE** oraz **HISTSIZE**. Najczęściej zmienne te zdefiniowane są następująco:

HISTFILE=~/.bash_history

HISTSIZE=1000
Aby obejrzeć na ekranie dostępne z historii polecenia wystarczy użyć polecenie:

history 10
gdzie **10** to ilość ostatnio używanych poleceń, które teraz chcemy wyświetlić.
Aby powtórzyć polecenie z listy wyświetlanej poleceniem **history,** wystarczy wpisać wykrzyknik, a po nim numer polecenia:

!3
Wcześniejsze polecenia można powtórzyć także bez znajomości numeru polecenia. Załóżmy, że jakiś czas temu wpisaliśmy **more /etc/passwd** , a teraz chcemy jeszcze raz ten plik obejrzeć, wtedy wystarczy napisać:

!more
Aby powtórzyć ostatnio wydane polecenie wystarczy napisac:

!!
Aby powtórzyć ostatnio wydane polecenie dodając do niego jakieś zmiany piszemy np.:

!! | more

Symbole wieloznaczne

Powłoka bash obsługuje następujące symbole wieloznaczne:
***** - które zastępuje dowolny znak lub ich ciąg
? - zastępuje jeden znak
[x-y] - zastępuje dowolny znak z zakresu od x do y
[x,y,z] - zastępuje znaki wymienione po przecinku

Symbole wieloznaczne wykorzystywane są najczęściej do wykonywania tego samego polecenia na różnych plikach, bądź katalogach:

Potoki

Łączą polecenia zwiększając możliwości oraz prędkość pracy. Potoki wykorzystywane są najczęściej w procesach przetwarzania danych np. gdy chcemy wyszukać jakieś konkretne pliki bądź przeglądać logi systemowe, uruchomione procesy itd. Zasada działania potoków polega na tym, że wyście poprzedniego polecenia stanowi wejście następnego:

ls –l | grep jakasnazwa

Zarządzanie zadaniami

Powłoka Bash oferuje możliwość uruchamiania programów w tle:

find / -name passwd > plik &
Program **find** uruchomi się w tle, a my na ekranie otrzymamy numer przypisanego mu procesu. W celu przeniesienia procesu na pierwszy plan wydajemy polecenie:

fg nr
gdzie **nr** to numer procesu.

Strumienie danych

Każdy uruchomiony w Linuxie proces skądś pobiera dane, gdzieś wysyła wyniki swojego działania i komunikaty o błędach. Tak więc procesowi przypisane są trzy strumienie danych:

- **stdin** (standard input) czyli standardowe wejście, skąd proces pobiera dane, domyślnie jest to klawiatura

- **stdout** (standard otuput) to standardowe wyjście, gdzie wysyłany jest wynik działania procesu, domyślnie to ekran

- **stderr** (standard error) standardowe wyjście błędów, tam trafiają wszystkie komunikaty o błędach, domyślnie ekran
BASH identyfikuje strumienie za pomocą przyporządkowanych im liczb całkowitych tak zwanych deskryptorów plików. I tak:

- **0** to plik z którego proces pobiera dane **stdin**

- **1** to plik do którego proces pisze wyniki swojego działania **stdout**

- **2** to plik do którego trafiają komunikaty o błędach **stderr**
Za pomocą operatorów przypisania można manipulować strumieniami, poprzez przypisanie deskryptorów: 0, 1, 2 innym plikom, niż tym reprezentującym klawiaturę i ekran.

* **P**rzełączanie standardowego wejścia :

Najpierw stwórzmy plik **lista** o następującej zawartości:

slackaware
redhat
debian
caldera

Użyjemy polecenia **sort** dla którego standardowym wejściem będzie nasz plik.

sort < lista

Wynikiem będzie wyświetlenie na ekranie posortowanej zawartość pliku lista:

caldera
debian
redhat
slackware

*** P**rzełączanie standardowego wyjścia:

ls -la /usr/bin > ~/wynik
Rezultat działania polecenia **ls -la /usr/bin** trafi do pliku o nazwie **wynik**, jeśli wcześniej nie istniał plik o takiej samej nazwie, to zostanie utworzony, jeśli istniał cała jego poprzednia zawartość zostanie nadpisana. Jeśli chcemy aby dane wyjściowe dopisywane były na końcu pliku, bez wymazywania jego wcześniejszej zawartości, stosujemy operator:

>>
*** P**rzełączanie standardowego wyjścia błędów:

cp plik /etc 2> error
W przypadku niepowodzenie operacji kopiowania komunikat o błędzie zapisany zostanie w pliku **error**

Aliasy

Aliasy powłoki umożliwiają określenie skróconych nazw poleceń, ewentualnie poleceń z uwzględnieniem argumentów. Składnia aliasu wygląda następująco:

alias name= 'zadanie'
gdzie **zadanie** określa polecenie, dla którego określamy alias, a **name**, to nazwa aliasu. Np.:

alias baza='cd /home/username/book'

Powyższe polecenie definiuje alias baza, który umożliwia szybkie przejście do katalogu **/home/username/book**. Teraz zamiast wydawać polecenie **cd /home/username/book** wystarczy napisać **baza**.

Aliasy usuwamy przy użyciu polecenia **unalias**. W przeciwnym razie są one widziane, dopóki nie opuścimy powłoki (to jest dopóki przynajmniej się nie wylogujemy). Gdy zależy nam alby aliasy były uaktywniane przy każdym logowaniu należy je umieścić w pliku **.bash_profile** w katalogu domowym. Składnia polecenia **unalias** wygląda następująco:

unalias name
gdzie **name** określa alias, który ma być usunięty.

Aby wyświetlić listę wszystkich dostępnych aliasów wystarczy wydać polecenie **alias**, bez jakichkolwiek parametrów.

Trzeba pamiętać że skróty alias maja pierwszeństwo przed poleceniami posiadającymi tę samą nazwę. Może to być wykorzystane, aby zapobiec niechcianym wywołaniom poleceń np.:

alias more=less
Od teraz każda próba wywołania polecenia **more** prowadzi do uruchomienia bardziej zaawansowanego, ale zastępczego programu **less**. Jeżeli potrzebujemy **more**, musimy określić kompletną ścieżkę dostępu do programu **more** czyli **/bin/more.**

Zmienne

Funkcje powłoki bash i wielu innych programów Linuxa są kontrolowane przez tzw. zmienne powłoki. Są one porównywalne do zmiennych języków programowania.
Przypisywanie zmiennych powłoki jest wykonywane za pomocą operatora przypisania:

MOJA=abc

W celu wyświetlenia zawartości zmiennej używamy polecenia echo ze zmienną poprzedzoną znakiem $:

echo $MOJA

Jeżeli zawartość zmiennej powłoki zawiera spacje lub inne znaki specjalne, to cały łańcuch znaków musi być ujęty w pojedyncze lub podwójne cudzysłowy w operacji przypisywania:

MOJA='jaka zmienna'

Do zmiennej można przypisywać wartość innej zmiennej w połączeniu z jakimś oryginalnym łańcuchem:

MOJA=3
MOJA2=$MOJA'nowe'$MOJA

Obliczenia ze zmiennymi mogą być wykonywane przy zastosowaniu zapisu w nawiasach kwadratowych:

A=$[3*4]

Zmienne środowiskowe są częścią środowiska systemowego i nie trzeba ich własnoręcznie definiować. Parametry przechowywane przez te zmienne są używane przez program powłoki przy obsłudze różnego rodzaju poleceń czy programów systemowych. Definiują one środowisko użytkownika, dostępne dla wszystkich procesów potomnych.

Najważniejsze predefiniowane zmienne:
BASH - interpretowane jako pełna nazwa pliku użyta do wywołania tego przebiegu bash
BASH_VERSION - interpretowane jako łańcuch opisujący wersję uruchomionego bash
EDITOR - określa domyślny edytor
HISTCMD - licznik historii bieżącego polecenia
HISTFILE - określa nazwę pliku historii
HISTSIZE - określa ilość poleceń przechowywanych w historii
HOSTNAME - nazwa hosta
HOSTTYPE - typ maszyny, na której jest wykonywany bash
MACHTYPE - łańcuch w pełni opisujący typ systemu, na którym jest wykonywany bash
MAIL - ścieżka gdzie przychodzą dla nas listy
MAILCHECK - czas między sprawdzeniami nowo otrzymanej poczty
MAILPATH - lista plików poczty, które mają być sprawdzane w poszukiwaniu nowo otrzymanych wiadomości
OLDPWD - poprzedni katalog roboczy

OSTYPE - system operacyjny, w którym jest wykonywany bash
PATH - ścieżka przeszukiwania
PIPESTATUS - zwiera listę kodów zakończenia z procesów w ostatnio wykonywanym potoku pierwszoplanowym
PPID - identyfikator procesu macierzystego powłoki
PS1 - znak zachęty pierwszego poziomu
PS2 - znak zachęty drugiego poziomu
PWD - bieżący katalog
RANDOM - generuje losowo całkowitą liczba z zakresu od 0 do 32767
SECONDS - zwraca liczbę sekund, jakie upłynęły od wywołania powłoki. Jeżeli do tej zmiennej zostanie przypisana wartość, to wartoć zwracana przez kolejne odwołania równa się liczbie sekund od czasu przypisania plus przypisana wartość
SHELL - określa ścieżkę do programu powłoki, jakiego używasz
TERM - nazwa terminala
UID - identyfikator bieżącego użytkownika

PATH

Wydając polecenie nie będące wbudowanym poleceniem basha, tylko odrębnym programem powłoka musi wiedzieć gdzie ma tego programu szukać. Z pomocą przychodzi zmienna **PATH**, w której zapisane są katalogi jakie maja zostać przeszukane w celu znalezienia interesującego nas programu. Zmienna **PATH** zdefiniowana może być następująco:

PATH=$HOME/bin:/usr/bin:.:/usr/X11R6/bin

Łańcuch na prawo od znaku równości jest wartością zmiennej środowiska **PATH**. Nazwy katalogów oddzielone są dwukropkami. Kropka oznacza katalog bieżący. Kiedy powłoka ma odszukać plik, przeszukuje katalogi wymienione w zmiennej środowiskowej **PATH**. Powłoka szuka po katalogach zmiennej **PATH** w kolejności, w jakiej tam występują. W przypadku dwóch programów o tej samej nazwie znajdujących się w różnych katalogach wykonany zostanie tylko pierwszy znaleziony. Jeżeli plik znajduje się w innym katalogu, którego nie ma na liście, a nie chcemy podawać pełnej ścieżki do tego pliku możemy ten katalog dodać do zmiennej **PATH** przy pomocy polecenia:

PATH=$PATH:/usr/games

PS1

Znak zachęty to tekst, który informuje użytkownika, że system czeka na wprowadzenie przez niego danych. Znak zachęty pierwszego poziomu **PS1** wyświetlany jest zaraz po zalogowaniu i informuje nas, że system czeka na wprowadzenie polecenia. Drugi poziom jest wyświetlany, gdy interpreter potrzebuje dodatkowych danych, które musimy wprowadzić. Jest on przechowywany w zmiennej **PS2** i standardowo ma postać **">".** Zmienne **PS1** oraz **PS2** można dowolnie konfigurować. Przykład Zmiennej **PS1**:

PS1="Tu znak zachety"

Nie jest to rozwiązanie wygodne. Lepiej utworzyć znak zachęty przedstawiający dynamicznie zmieniające się dane (np. czas, bieżący katalog, numer aktualnego polecenia itp.). Standardowo zmienna **PS1** ma postać:

PS1="[\u@\h \W]\\$ "

Pierwszy znak **[** - jest znakiem statycznym, dzięki niemu dane przedstawione są w estetycznej formie, kod **\u** wyświetla identyfikator użytkownika, **@** to następny znak statyczny, **\h** pokazuje nazwę komputera, a **\W** wywietla podstawową nazwę katalogu. Następnie znak **]** zamyka nawias, a po nim następuje kod **\$**, który wywietla znak **#** gdy jestemy zalogowani jako root, lub **$** gdy jako zwykły użytkownik. Po przetworzeniu zapis

PS1="[\u@\h\W]\\$"

wygląda następująco:

[root@soho/root]#

Oto lista kodów specjalnych, które mogą zostać użyte do konfiguracji znaku zachęty:

\! - numer polecenia historii,
\# - numer aktualnego polecenia,
\$ - # dla root`a, $ dla zwykłego użytkownika
**** - znak \
\d - data
\t - czas
\h - nazwa komputera
\n - przechodzi do nowego wiersza
\s - nazwa powłoki
\u - login użytkownika
\W - wyświetla podstawową nazwę bieżącego katalogu
\w - wyświetla nazwę bieżącego katalogu

Zmienne wewnętrzne są definiowane przez system. W przeciwieństwie do zmiennych środowiskowych - nie można zmieniać ich wartości. Można je wykorzystać do sterowania wykonaniem programu. Oto niektóre z nich:

$# - Liczba parametrów pozycyjnych przekazanych do programu
$? - Kod zakończenia ostatniego polecenia lub programu powłoki wywoływanego z programu powłoki (wartość zwrócona)
$0 - Nazwa programu powłoki
$* - Pojedynczy tekst zawierający wszystkie argumenty podane przy wywołaniu programu
$1 , $2 ... - Kolejne parametry programu
$@ - Lista parametrów pozycyjnych przekazana do programu powłoki w postaci pojedynczego łańcucha znaków

Przykładowy program o nazwie **MOJ** wykorzystujący zmienne wewnętrzne:

```
#Program testowy
echo "Liczba parametrów: $#"
echo "Nazwa programu: $0"
echo "Parametry w postaci pojedynczego ciągu znaków: $*"
```

Uruchomienie programu:

./MÓJ parametr1 parametr2

Na ekranie pojawi się:

Liczba parametrów: 2
Nazwa programu: MOJ

Parametry w postaci pojedynczego ciągu znaków: parametr1 parametr2

Zmienne definiowane w powłoce są dla niej lokalne. Nie można się do nich odwoływać z innych powłok (zmienna jest ukryta w powłoce). W powłoce BASH zmienne środowiskowe są eksportowane. Oznacza to, że po wyeksportowaniu kopia zmiennej środowiskowej jest dostępna we wszystkich podpowłokach. Jeżeli zmienna o nazwie **MOJA** zostanie wyeksportowana automatycznie definiowana jest kopia w każdej podpowłoce:

export MOJA=abc

Każda nowa powłoka będzie miała własną kopię zmiennej. Jeżeli zdefiniowaliśmy więcej zmiennych, to można je eksportować następująco:

export ZMIENNA1 ZMIENNA2 ZMIENNA3 ZMIENNA4

source plik [argumenty] - Odczytuje i wykonuje polecenia z żądanego pliku w aktualnym środowisku powłoki i zwraca kod zakończenia ostatniego wykonanego polecenia z tego pliku

alias [-p] [nazwa[=wartość] ...] - Definiowany jest alias (synonim) dla każdej nazwy, dla której podano wartość.

bg [nr] - Wznawia w tle zawieszone zadanie o numerze **nr**.

builtin [argumenty] - Wykonuje zadane polecenie wbudowane powłoki, przesyłając mu argumenty i zwraca jego kod zakończenia.

cd [-LP] [katalog] - Zmienia bieżący katalog roboczy na katalog.

command [-pVv] polecenie [arg ...] - Uruchamia polecenie z argumentami zakazując zwykłego wyszukiwania funkcji przez powłokę.

compgen [opcja] [słowo] - Tworzy możliwe dopasowania uzupełnień dla słowa zgodnie z opcjami, które mogą być dowolnymi z opcji akceptowanych przez wbudowane polecenie complete, za wyjątkiem -p i -r, i wypisuje dopasowania na standardowe wyjście.

typeset [-afFirx] [-p] [nazwa[=wartość]] - Deklaruje zmienne i/lub nadaje im atrybuty.

dirs [-clpv] [+N] [-N] - Bez opcji wyświetla listę aktualnie zapamiętanych katalogów.

disown [-ar] [-h] [zadanie ...] - Bez opcji, każde z podanych zadań usuwane jest z tablicy zadań aktywnych.

echo [-neE] [argument ...] - Wyświetla argumenty, rozdzielone spacjami, zakończone znakiem nowej linii.

exec [-cl] [-a nazwa] [polecenie [argumenty]] - Jeżeli podano polecenie, zastępuje ono powłokę.

hash [-r] [-p plik] [nazwa] - Dla każdej nazwy określana i zapamiętywana jest pełna nazwa plikowa polecenia wyszukanego w katalogach **$PATH** .

help [-s] [wzorzec] - Wyświetla pomocne informacje o poleceniach wbudowanych.

history [n] - Bez żadnych opcji, wyświetla listę historii poleceń z numerami wierszy.

jobs [-lnprs] [zadanie ...] - Podaje aktywne zadania.

kill [-s sigspec | -n signum | -sigspec] [pid | jobspec] - Wysyła sygnały określone przez **sigspec** lub **signum** do procesu określonego przez **pid** lub **jobspec**.

let arg [arg ...] - Każdy argument jest wyrażeniem arytmetycznym, jakie ma zostać zinterpretowane.

logout - Kończy pracę powłoki zgłoszeniowej.

popd [-n] [+n] [-n] - Usuwa pozycje ze stosu katalogów.

printf format [argumenty] - Zapisuje sformatowane argumenty na standardowe wyjście przy pomocy żądanego formatu.

pushd [-n] [katalog] - Dodaje katalog na wierzchołek stosu katalogów, albo obraca stos, czyniąc nowy wierzchołek stosu bieżącym katalogiem roboczym.

readonly [-apf] [nazwa ...] - Podane nazwy zmiennych oznaczane są jako tylko do odczytu.

set [--abefhkmnptuvxBCHP] [-o opcja] [arg ...] - Bez opcji, wyświetlane są nazwa i wartość każdej ze zmiennych powłoki, w formacie który może być ponownie wykorzystany jako wejście.

suspend [-f] - Zawiesza wykonywanie tej powłoki do otrzymania przez nią sygnału **SIGCONT**.

times - Drukuje sumaryczne czasy użytkownika i systemu dla powłoki i procesów z niej uruchomionych

type [-atp] nazwa [nazwa ...] - Bez opcji wskazuje, jak powinna być interpretowana każda z nazw, jeśli zostanie użyta jako nazwa polecenia.

ulimit [-SH [limit]] - Zapewnia kontrolę nad zasobami dostępnymi powłoce i procesów jakie ona uruchamia, oczywiście na systemach umożliwiających taką kontrolę.

unalias [-a] [nazwa ...] - Usuwa każdą z nazw z listy zdefiniowanych aliasów.

unset [-fv] [nazwa ...] - Dla każdej nazwy, usuwa odpowiadającą jej wartość lub funkcję.

wait [n] - Czeka na określony proces i zwraca jego kod zakończenia.