

Wstęp do systemu operacyjnego UNIX

Laboratorium 7:

Interpretery poleceń systemu Unix

Interpreter poleceń (shell), stanowiący interfejs między użytkownikiem, a jądrem systemu nie jest integralną częścią systemu operacyjnego UNIX. Każdy, kto zna prorokół komunikacji z jądrem może napisać swój własny interpreter. Stąd ich mnogość. Faktycznie znaczenie mają jedynie dwie rodziny, a mianowicie `sh` oraz `csh`. Kryterium podziału stanowi składnia języka programowania dostępnego w danym shellu. Każdy interpreter poleceń, udostępniający linię komend pracuje cyklicznie, a w każdym kroku można wyróżnić 4 kroki:

1. Oczekiwanie na wprowadzenie komendy i naciśnięcie klawisza `Enter`.
2. Analiza linii komend. Jeśli linia jest poprawna składniowo shell przystępuje do odszukania odpowiedniej komendy. W przypadku błędu składniowego pojawia się stosowny komunikat. Komenda do wykonania może być poleceniem wbudowanym w interpreter, aliasem lub poleceniem zewnętrznym. Jeśli nie zostanie znalezione pojawia się komunikat o błędzie. Zwracamy uwagę iż nieomal wszystkie polecenie systemów typu UNIX są normalnymi programami, liczba poleceń wbudowanych ogranicza się do kilku.
3. Znalaziona komenda jest kierowana do wykonania przez jądro. Jeśli komenda została skierowana do wykonania w tle, to interpreter nie oczekuje na wyniki wykonania tylko przechodzi do punktu 1.
4. Interpreter oczekuje na zakończenie wykonania komendy przez jądro i wyniki kieruje na standardowe wyjście i wyjście diagnostyczne. Następnie przechodzi do kroku 1.

Koniec pracy takiego interpretera ma miejsce po pojawieniu się na jego wejściu (w linii komend) znaku `Control-d` lub po wykonaniu komendy `logout` lub `exit`.

W zależności od sposobu uruchomienia oraz trybu pracy rozróżnia się trzy rodzaje interpreterów:

1. Logujący (zgłoszeniowy) uruchomiony przez program **login** w momencie podłączenia użytkownika do systemu.
2. Interakcyjny to taki, który został uruchomiony z linii komend jako najzwyczajniejszy proces.
3. Interpretujący to uruchomiony przez bieżący interpreter poleceń do zinterpretowania skryptu (programu dla interpretera poleceń).

Interpretery te różnią się także plikami, które interpretują w momencie uruchamiania. Różne jest zatem środowisko pracy każdego z rodzajów, a co za tym idzie możliwości. Stąd może zdarzyć się, że interpreter logujący uruchomi nam pewien program, a interaktywny wypisze komunikat o nie znalezieniu komendy. O kolejności interpretowania plików przez interpretery można przeczytać w manualu. Przykładowo interpreter `bash` logujący czyta i wykonuje polecenia z pliku `/etc/profile`, jeśli ten plik istnieje. Po przeczytaniu tego pliku szuka plików `~/.bash_profile`, `~/.bash_login` oraz `~/.profile` we wskazanej kolejności oraz czyta i wykonuje polecenia z pierwszego pliku, który istnieje i który można przeczytać Kiedy interpreter logujący kończy pracę (użytkownik odłącza się od systemu) `bash` czyta i wykonuje polecenia z pliku `~/.bash_logout`, jeśli ten plik istnieje. Pozostałe rodzaje interpretera `bash` (interaktywny i interpretujący) czytają i wykonują polecenia z pliku `~/.bashrc`, jeśli ten istnieje.

Logujący interpreter poleceń `tcsh` czyta i wykonuje polecenia z pliku `/etc/csh.cshrc` i `/etc/csh.login`. Po przeczytaniu ich zawartości szuka plików `~/.tcshrc` oraz `~/.cshrc` we wskazanej kolejności i czyta oraz wykonuje komendy z pierwszego znalezione. Następnie uzupełnia listę historii komend z pliku `~/.history` jeśli plik ten istnieje. Następnie poszukuje pliku `~/.login` i jeśli istnieje wykonuje z niego komendy. Podczas odłączania się od systemu, gdy shell logujący kończy pracę poszukuje on pliku `~/.logout` i jeśli plik ten istnieje wykonuje zawarte w nim komendy. Pozostałe rodzaje interpreterów poleceń `tcsh` poszukują i interpretują jedynie zawartość plików `/etc/csh.cshrc` i `~/.tcshrc` lub `~/.cshrc`.

Należy zwrócić uwagę, iż wszystkie komendy i polecenia, których charakter jest jednokrotny i należy wykonywać je jedynie przy podłączaniu się do systemu lub odłączaniu się od niego należy umieszczać w plikach interpretowanych przez shelle logujące. Ustawienia zmiennych środowiskowych, jak np. ścieżek dostępu

przeszukiwanych przez interpretery poleceń w celu odnalezienia komendy do uruchomienia powinny znaleźć się w plikach interpretowanych przez każdy rodzaj interpretera.

0. Podłącz się do systemu jako użytkownik `test`. Jeśli w systemie użytkownik ten nie jest zdefiniowany lub nie jest znane hasło, to podłącz się jako użytkownik `root` i rozwiąż ewentualne problemy.

Do wypisywania wartości zmiennych zdefiniowanych w środowisku interpretera poleceń (np. zmiennych środowiskowych) lub dowolnych napisów służy komenda `echo`. Jeśli zechcemy wypisać sam tekst składający się z kilku wyrazów oddzielonych znakiem białym to tekst należy ująć w apostrofy. W tym przypadku jest obojętne, czy użyjemy pojedynczych czy podwójnych:

```
1 [bory@thorin bory]$ echo "kilka wyrazow oddzielonych spacjami"
2 kilka wyrazow oddzielonych spacjami
3 [bory@thorin bory]$ echo 'kilka wyrazow oddzielonych spacjami'
4 kilka wyrazow oddzielonych spacjami
```

Jeśli natomiast między wyrazami ma się pojawić wartość zmiennej środowiskowej, to rodzaj znaczników tekstu nie jest bez znaczenia:

```
1 [bory@thorin bory]$ echo 'wartosc zmiennej MAIL: $MAIL'
2 wartosc zmiennej MAIL: $MAIL
3 [bory@thorin bory]$ echo "wartosc zmiennej MAIL: $MAIL"
4 wartosc zmiennej MAIL: /var/spool/mail/bory
```

Jak widać pojedyncze apostrofy „zamykają” tekst całkowicie przed ciekawością interpretera poleceń. Podwójne cudzysłowia dopuszczają interpretację pewnych znaków specjalnych. Oprócz znaku \$, będącym operatorem zwracającym wartość zmiennych środowiskowych, do znaków tych należą również odwrócone apostrofy, umożliwiające wykonywanie komend i zwracanie wyniku ich działania np. w komendzie `echo`:

```
1 [bory@thorin prof]$ echo "biezacy katalog : 'pwd'"
2 biezacy katalog : /home/bory/prof
3 [bory@thorin prof]$ echo 'biezacy katalog : 'pwd''
4 biezacy katalog : 'pwd'
```

Polecenie `echo` może posłużyć do sprawdzenia w jaki sposób zostaną zinterpretowane znaki uogólniające powłoki. np:

```
1 [bory@thorin prof]$ echo /bin/l*
2 /bin/ln /bin/login /bin/ls
3 [bory@thorin prof]$ echo /bin/l?
4 /bin/ln /bin/ls
5 [bory@thorin prof]$ echo {ala,ela,ola}" ma kota  "
6 ala ma kota  ela ma kota  ola ma kota
7 [bory@thorin prof]$ echo /bin/[e-m]?
8 /bin/ed /bin/ex /bin/ln /bin/ls /bin/mt /bin/mv
```

1. Wypisz wartość zmiennej środowiskowej `SHELL`. Mówi ona o tym jaki interpreter poleceń jest interpreterem logującym. Wartość tej zmiennej jest ustawiana na podstawie zawartości pliku `/etc/passwd`. Jeśli jest inna niż `*.bash`, to uruchom interaktywny interpreter `bash`.

Zmienne środowiskowe oraz programowe wchodzą w skład środowiska każdego procesu z procesem interpretera poleceń włącznie. Proces potomny dziedziczy po procesie rodzica środowisko wraz ze zmiennymi. Aby te jednak mogły zostać odziedziczone muszą zostać wyeksportowane. W rodzinie shelli `sh` zmienne definiuje się według schematu : `ZMIENNA=wartość`, zaś eksportuje: `export ZMIENNA` lub też z jednoczesnym nadaniem wartości: `export ZMIENNA=wartość`. W rodzinie `csh` definiujemy i eksportujemy zmienne przy pomocy komendy `setenv`: `setenv ZMIENNA wartość`.

2. Zdefiniuj i wyeksportuj zmienną o nazwie `TEST` nadając jej wartość `/home/test/bin`. Uruchom interpreter interaktywny. Sprawdź, czy odziedziczył on zmienną `TEST` po procesie rodzica. Zakończ działanie interpretera interaktywnego.
3. Sprawdź czy można odziedziczyć zmienną bez nadanej wartości.
4. Uruchom interaktywny interpreter `tcsh`. Sprawdź, czy odziedziczył on zmienną `test` po procesie rodzica (interpreter `bash`). Zakończ pracę interaktywnego interpretera `tcsh`.

Wszystkie zmienne, które tworzą środowisko bieżącego interpretera poleceń można w rodzinie interpreterów `sh` wylistować przy pomocy komendy `set`. Zmienne, które zostaną odziedziczone przez proces potomny listujemy komendą `env`. Zmienne środowiskowe usuwamy przy pomocy komendy `unset` podając jako argument wywołania nazwę zmiennej.

5. Sprawdź, czy faktycznie brak wydania komendy `export` powoduje, że dowolna zmienna będzie dostępna jedynie w środowisku bieżącego shella (zdefiniuj dowolną zmienną np. `XYZ=6` bez użycia komendy `export` i sprawdź, czy listuje ją komenda `set`, a nie widzi jej komenda `env`. To samo powtórz wykorzystując komendę `export`).

Interpretery poleceń dopuszczają w standardowej konfiguracji komendy o długości do 1024 znaków. Pisanie tak długich komend w jednej linii ekranu jest niewygodne. Interpretery poleceń dopuszczają „łamanie” wiersza poleceń. Jeśli na końcu linii komend pojawi się znak odwrotnego ukośnika `\` oznacza on, że wiersz polecenia będzie kontynuowany. Pierwsza linia bez znaku `\` na końcu oznacza koniec komendy i interpreter rozpoczyna jej obróbkę:

```

1 [bory@thorin bory]$ ls -l\
2 > | grep txt
3 -rw-rw-r-- 1 bory bory 1992 lip 11 15:35 kki.txt
4 -rw-rw-r-- 1 bory bory 3708 lip 30 2002 pub.txt
5 -rw-rw-r-- 1 bory bory 962 lip 26 2002 zak.txt

```

Jak widać na powyższym listingu pojawiły się dwa znaki zachęty. Pierwszy ma postać z nazwą użytkownika, nazwą maszyny i bieżącego katalogu. Drugi to znak `>`. Ich postaci definiują zmienne środowiskowe o nazwach odpowiednio `PS1` i `PS2`.

6. Sprawdź wartości zmiennych środowiskowych `PS1` i `PS2`.
7. Ustaw wartość zmiennej środowiskowej `PS1` tak, aby znak zachęty zawierał: nazwę użytkownika, pełną nazwę hosta, czas w formacie 24-godzinny, bieżący katalog i numer komendy. Odpowiednie wartości znajdź w manualu dla interpretera poleceń `bash` (poszukiwany napis `prompting`). Zmiennej `PS2` nadaj wartość taką, aby miał on postać: `dalej->`.
8. Zapisz zaproponowane postaci do odpowiedniego pliku konfiguracyjnego, tak aby były zauważane przez wszystkie rodzaje interpreterów. Sprawdź poprawność ustawień uruchamiając interpretery interaktywne.

Wyszukując komendę do wykonania, interpreter poleceń korzysta z wartości zmiennej `PATH`. Wartość tej zmiennej stanowią ścieżki dostępu do katalogów zawierających programy do wykonania oddzielone od siebie dwukropkiem („;”). Zmienna przeglądana jest od strony lewej do prawej i przekazywana do wykonania jest pierwsza napotkana komenda. Ze względów bezpieczeństwa na liście tej nie umieszcza się kropki `.` oznaczającego katalog bieżący. Doklejanie nowych zmiennych do wartości zmiennej `PATH` zazwyczaj odbywa się według schematu: do starej wartości dołącz nową i podstaw z powrotem pod zmienną:

```

1 [bory@thorin bory]$ echo $PATH
2 /usr/local/bin:/bin:/usr/bin:/opt/mpich/mpich-1.2.5/bin
3 [bory@thorin bory]$ export PATH=$PATH:~
4 [bory@thorin bory]$ echo $PATH
5 /usr/local/bin:/bin:/usr/bin:/opt/mpich/mpich-1.2.5/bin:/home/bory

```

9. W katalogu osobistym użytkownika `test` utwórz podkatalog `programs`. Przejdź do katalogu `/programs`. Z katalogu `/bin` skopiuj do katalogu bieżącego plik `ls` nadając mu nazwę `listuj`. Przejdź do

katalogu osobistego. Wywołaj komendę `listuj`. Jaki komunikat pojawił się na ekranie? Sprawdź komendą `which` podając jako argument nazwę komendy `listuj`, jaka jest ścieżka dostępu do tej komendy.

10. Zmodyfikuj wartość zmiennej `PATH` dopisując do niej ścieżkę dostępu do podkatalogu `programs` z katalogu domowego. Użyj teraz komendy `listuj`. Przy pomocy komendy `which` sprawdź, który plik `listuj` jest brany pod uwagę w chwili uruchamiania komendy `listuj`.

W większości interpreterów poleceń pamiętane są komendy wywoływane w przeszłości przez danego użytkownika. Zapisywane są one w pliku do którego ścieżka dostępu jest określona przez wartość zmiennej środowiskowej `HISTFILE`.

11. Gdzie znajduje się i jak się nazywa plik z historią poleceń. Użyj komendy `history` do wypisania wszystkich wywołanych komend.

Komendy z pliku historii mogą być przywoływane do linii komend klawiszami strzałkowymi. Mogą być również uruchamiane z wykorzystaniem znaku wykrzyknika jako pierwszego w linii komend i podaniu po nim numeru komendy z listy lub pierwszych jej liter:

```
1 [bory@thorin bory]$ !1010
2 echo $HISTFILE
3 /home/bory/.bash_history
4 [bory@thorin bory]$ !echo
5 echo $HISTFILE
6 /home/bory/.bash_history
```

Liczba pamiętanych komend jest określona wartością zmiennej środowiskowej `HISTSIZE`. Jeśli liczba komend wydanych przez użytkownika osiągnie wartość zmiennej środowiskowej `HISTSIZE`, najstarsze komendy są z listy usuwane, a nowe dodawane (kolejka FIFO).

12. Ustaw wartość zmiennej środowiskowej `HISTSIZE` na 10. Sprawdź zawartość pliku komend przy pomocy komendy `history`. Powtórz dowolną z komend używając znaczka „!”.

Podczas pracy w systemie często tworzymy długie linie komend, z wieloma przekierowaniami i filtrami. Dodatkowo, wywołania tych komend powtarzane są wielokrotnie. Poszukiwanie takich komend w pliku historii lub przywoływanie ich klawiszami strzałkowymi bywa uciążliwe. Pewnym ułatwieniem w takiej sytuacji jest mechanizm aliasów. Polega on na nadaniu krótkiej nazwy (przezwoła) złożonej komendzie. Definicje aliasów można zapisać w pliku konfiguracyjnym shella dzięki czemu będą one ustawiane automatycznie podczas uruchamiania shella. Dla interpreterów poleceń rodziny `sh` składnia polecenia `alias` definiującego aliasy wygląda następująco:

alias nazwa_krótko = komenda

Przykładowo:

```
1 [bory@thorin bory]$ alias ll='ls -l'
2 [bory@thorin lab3]$ ll
3 razem 12
4 drwxrwxr-x  2 bory  bory  4096 sie  5 01:50 text1
5 drwxrwxr-x  2 bory  bory  4096 sie  5 01:44 text2
6 drwxrwxr-x  2 bory  bory  4096 sie  5 02:13 text3
```

Nazwa krótka może być dowolna. W szczególności może być nazwą innej komendy. Wówczas w prostym wywołaniu z linii komend uruchomiony zostanie alias, gdyż wynika to z kolejności przeszukiwania poleceń przez shell. Komenda może być komendą wbudowaną interpretera, komendą zewnętrzną lub aliasem. W naszym przykładzie:

```
1 [bory@thorin lab3]$ which ll
2 alias ll='ls -l'
3 /bin/ls
```

Alias można usunąć (oddefiniować) komendą `unalias`.

13. Sprawdź, czy istnieje komenda `ln` i jeśli istnieje i jest aliasem, to oddefiniuj go. Przy pomocy komendy `alias` zdefiniuj komendę `ln` listującą zawartość bieżącego katalogu w formacie długim z podaniem numeru i-węzła i stronicowaniem komendą `less`. Użyj zdefiniowanego aliasu do wylistowania zawartości katalogu `/tmp`. Usuń alias `ln`. Spróbuj wywołania `ln` ponownie. Jaki komunikat pojawił się na ekranie?

Każdy proces uruchomiony w systemie Unix otrzymuje deskryptory do trzech otwartych plików. Tymi plikami są: standardowe wejście - plik o deskrytorze 0, standardowe wyjście - deskryptor 1 oraz wyjście diagnostyczne - deskryptor 2. Standardowe wejście skojarzone jest z urządzeniem reprezentującym klawiaturę. Najczęściej standardowe wyjście oraz wyjście diagnostyczne skierowane są do urządzenia reprezentującego terminal. Oczywiście istnieje możliwość przekierowywania plików o deskryptorach 0, 1 i 2 do innych urządzeń lub plików. W przypadku standardowego wejścia przekierowanie w shellach obu rodzin jest takie samo i ma postać:

komenda < nazwa_pliku

Podobnie jest w przypadku standardowego wyjścia:

komenda > nazwa_pliku

W rodzinie shelli `sh` może przyjąć postać zawierającą jawnie podany numer przekierowywanego pliku. Dla standardowego wyjścia będzie to postać:

komenda 1> nazwa_pliku

14. Przejdź do katalogu `/usr`. Przy pomocy komendy `find` znajdź w tym katalogu wszystkie te pliki, których nazwy kończą się na `.c`. Ile razy podczas przeszukiwania pojawił się komunikat o błędzie.

Wskazówka: w manualu do interpretera bash sprawdź jak przekierować wyjście diagnostyczne do pliku, a następnie policz wiersze w tym pliku.

15. Uruchom interakcyjny interpreter `tcsh`. Wykonaj poprzednie ćwiczenie w tym interpreterze.

Wskazówka: przekierowując w interpreterze poleceń `tcsh` strumienie `stdout` oraz `stderr` utwórz w katalogu `/tmp` plik `all.txt` zawierający znaki z obu strumieni oraz plik `out.txt` zawierający znaki ze strumienia `stdout`. Komendą `diff` porównaj zawartość obu plików. W wyniku porównania otrzymasz jedynie to, co zostało skierowane do strumienia `stdout`.

16. Odszukaj w dokumentacji interpretera poleceń `tcsh` jakie znaczenie ma zmienna `ignoreeof`.

Pracując w `tcsh` zkonfiguruj go w taki sposób, aby zakończenie pracy interpretera było możliwe po 5-krotnym naciśnięciu sekwencji `Control-d`. Zakończ w ten sposób pracę interaktywnego interpretera `tcsh`.

17. Uruchom w trybie interaktywnym interpreter poleceń `bash`. Wykonaj poprzedni punkt dla tego interpretera.

Na zakończenie kwestia związana z automatycznym odłączaniem użytkowników przejawiających oznaki bezczynności. W każdym interpreterze poleceń istnieje zmienna, której wartość oznacza liczbę sekund, przez jaką interpreter czeka na aktywność w wierszu poleceń przed automatycznym wylogowaniem użytkownika. Zmienna ta nie pomoże oczywiście jeśli użytkownik uruchomił z wiersza poleceń jakiś program, natomiast załatwi sprawę użytkowników – zapominałskich (patrz również komenda `w`). Zmienna ta występuje w każdym rodzaju interpretera poleceń. Dla interpretera `bash` nazywa się `TMOU`, a jej działanie przebiega następująco:

```
1 [bory@thorin bory]$ TMOU=100
2 [bory@thorin bory]$ timed out waiting for input: auto-logout
3 [bory@thorin bory]$
```

18. Nadaj wartość zmiennej `TMOU` taką, aby po 2 minutach bezczynności w linii komend shell automatycznie zakończył pracę. Sprawdź poprawność ustawienia.

Jednym z zadań administratora jest przygotowywanie konfiguracji systemu w taki sposób aby użytkownik mógł pracować bez konieczności poznawania szczegółów konfiguracji systemu. W wielu sytuacjach sprowadza się to do ustawienia/modyfikacji zmiennych środowiskowych w ustawieniach globalnych obowiązujących wszystkich użytkowników.

19. Zmodyfikuj plik(i) startowe interpretowane przy każdym uruchomieniu interpretera poleceń `bash` w taki sposób aby każdy użytkownik podłączając się do systemu miał zdefiniowaną zmienną środowiskową o nazwie `STUDENT` z wartością równą „KI_AGH”. W których plikach należy dokonać takiej zmiany? Testy przeprowadź podłączając się jako użytkownik `test` bezpośrednio z konsoli.
20. Zdefiniuj w systemie użytkownika `tester` w taki sposób aby domyślnym jego interpreterem poleceń był `tcsh`. Wykonaj punkt poprzedni dla interpretera poleceń `tcsh`. Do testów wykorzystaj użytkownika `tester`.