Linux
Polecenia

Opracował: Andrzej Nowak

Procesy i shell. Polecenia ps, sleep, exit, jobs, bg, fg, top, kill, metaznak ”&” i “>>”, bash, tcsh, which, type, whereis, touch.

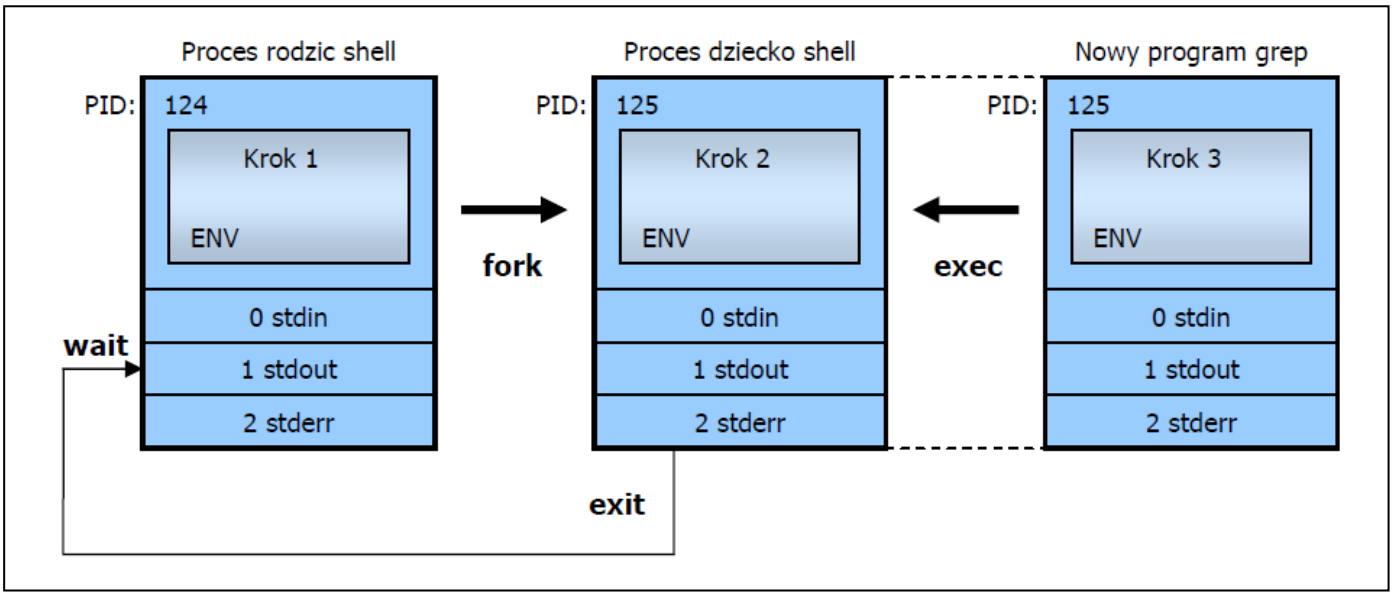
Proces jest programem, który jest wykonywany i który jest identyfikowany przez swój unikalny numer PID. Pierwszy wykonywany proces w systemie to init z PID nr 1. Jądro SO kontroluje i zarządza procesami. Proces składa się z:

- wykonywanego programu,
- danych i stosu,
- wskaźnika stosu i programu,
- rejestrów,
- informacji niezbędnych do wykonania (uruchomienia programu).

Wywołany shell również jest procesem. Shell może również tworzyć procesy poprzez wywołania systemowe (ang. system calls) do jądra SO. Istnieje pewna liczba wywołań systemowych służących do tworzenia, wykonywania i zakończenia procesu. Są to odwołania:

- **fork**
- **wait**
- **exec**
- **exit**

Dodatkowo każdy proces zawiera informacje, którego użytkownika jest własnością, czyli kto go uruchomił.



fork - powoduje utworzenie nowego procesu, tzw. procesu dziecko (potomnego) będącego kopią procesu, który spowodował odwołanie systemowe i który jest nazywany procesem rodzicem (macierzystym).

wait - powoduje przejście procesu typu rodzic w stan oczekiwania.

exec - proces dziecko, po odszukaniu ścieżki na dysku, gdzie znajduje się wykonywalny program odpowiadający poleceniu, które nie jest wbudowane w jądro (np. ls), wydaje exec, który powoduje załadowanie do pamięci, na miejsce procesu - dziecko tego nowego

programu, który sam staje się procesem - dzieckiem i zaczyna być wykonywany.

exit - wykonywany proces - dziecko może zakończyć się w dowolnej chwili poprzez wykonywanie/wywoływanie exit.

- gdy proces - dziecko kończy się exit powoduje wysłanie sygnału i oczekuje od rodzica akceptacji swojej przyczyny zakończenia (statusu wyjścia),
- status wyjścia jest liczbą z zakresu 0 – 255,
- wartość 0 oznacza, że program zakończył się pomyślnie,
- wartość != 0 oznacza, że nastąpiło awaryjne wyjście z programu,
- shell posiada wbudowaną zmienną ($ status w shellu C, $? w shellu Bourne'a i Korn'a), która pamięta status wyjścia ostatniego wykonywanego polecenia.

Jeżeli z jakichś powodów "rodzic" nie czeka na zakończenie procesu "dziecko", to zostaje ono zawieszone dopóki "rodzic" nie wywoła wait lub "rodzic" umrze. Jeżeli rodzic umrze przed dzieckiem, proces init adoptuje proces – dziecko.

Każdy proces ma swój PID

```
$ ps
$ ps -f
```

Pełna informacja o procesach

UID – identyfikator użytkownika (kto uruchomił proces)
PID – identyfikator procesu
PPID – identyfikator procesu macierzystego danego procesu (id procesu, który go uruchomił).
C – wykorzystanie procesora w jednostce czasu
STIME – czas utworzenia procesu (kiedy się rozpoczął)
TTY – terminal
CMD – polecenie, które uruchomiło proces

```
$ ps -e | more
```

Pokaże wszystkie procesy

```
$ ps ax
$ ps -u
```

Podaje dodatkowe informacje o procesach, podobne do –f. Po wpisaniu tego polecenia pojawia się rozszerzony opis procesów. Warto zwrócić uwagę na kolumnę STAT. Wspomniana kolumna może przybrać: (R) - proces działający, (S) - uśpiony, (T) - zatrzymany, (Z) - zombie.

Wpisz w drugiej utworzonej powłoce:

```
$ ps
$ bash
$ ps -f
$ tcsh
$ ps -f
$ exit
$ ps
$ exit
$ ps
```

Polecenie kill służy do zakończenia procesu, którego jesteśmy użytkownikiem

```
$ kill PID
$ kill -9 PID
```

Usuń bezwarunkowo proces PID

```
$ kill -l
```

Wyświetli listę wszystkich sygnałów

```
$ man 7 signal
```

Inne polecenie wyświetlające listę procesów:

```
$ top
```

Lista procesów najbardziej absorbujących pamięć

Metaznak & (przetwarzanie procesu "w tle")

```
$ date; who | wc
$ (date; who) | wc
$ sleep 10
$ sleep 10 <Enter>
date <Enter>
$ sleep 20& <Enter>
$ date <Enter>
```

```
$ ps
$ who
$ ps
$ ps
```

```
$ (sleep 15; ls)& <Enter>
$ date <Enter>
$ ps
$ ps
```

```
$ (sleep 10; who)& date
$ (sleep 15; ls)& (sleep 10; who)
$ (sleep 15; ls)& (sleep 10; who)&
```

Jeżeli zapomnieliśmy wysłać proces w tło, zawsze można zrobić to później.

```
$ sleep 100
```

Naciśnij Ctrl + Z

```
$ jobs
$ bg
$ fg
```

Tworzenie pustego pliku

Utwórz katalog **lab_2** i przejdź do niego.

```
$ touch pusty
$ ls -l
```

Status wyjścia wykonania polecenia

```
$ cp pusty
$ echo $?
    1
$ cp pusty p_ty
$ echo $?
    0
```

Można sprawdzić czy dane polecenie jest wykonywalnym programem

```
$ type cat
$ which cat
$ which ls
$ which cd
```

```
$ whereis ls
```

Kilka poleceń można wykonać z pojedynczej linii poleceń

```
$ date; ls
$ date;\
ls
$ cd lab_2
```

Wynik wykonania polecenia można skierować do pliku zamiast na standardowe wyjście

```
$ cat p_ty
$ date > p_ty
$ cat p_ty
```

Polecenie `cat` można wykorzystać do połączenia kilku plików w jeden plik

```
$ cat > p1
```

Co by tutaj napisać.

Naciskamy Ctrl + C

```
$ ls
$ cat p1
```

Utwórzmy kilka plików:

```
$ date > data
$ w > kto
$ ls > lista
$ ls
```

Łączenie plików

```
$ cat p1 data kto > razem
$ cat razem
```

Dołączanie do pliku

```
$ cat data lista >> razem
$ cat razem
```

Ale!

```
$ cat data lista > razem
$ cat razem
```