# PROCESY SYSTEMOWE

## Przegląd procesów

Każdy program uruchomiony w systemie Linux tworzy proces. Kiedy logujemy się do systemu i uruchamiana jest powłoka jest ona procesem. Kiedy uruchamiamy polecenie lub otwieramy aplikację, uruchamiamy proces. Procesem jest więc każdym program działający w systemie.

System uruchamia także procesy zwane demonami. Demony są procesami uruchamianymi w celu wykonania konkretnego zadania. Na przykład demon logowania o nazwie dtlogin zapewnia graficzny interfejs dla wprowadzenia loginu i hasła użytkownika.
Każdemu procesowi zostaje przypisany unikalny numer identyfikacyjny (PID), który jest używany przez jądro systemu do śledzenia i zarządzania procesami. Dla użytkownika, numer PID jest wykorzystywany do identyfikacji i kontroli procesów.

## Identyfikatory UID oraz GID

Jądro systemu określa uprawnienia dla procesu na podstawie informacji o właścicielu procesu. Do tego celu przechowywane są identyfikatory UID oraz GID.
Identyfikatory UID i GID procesu mają te same wartości co identyfikatory UID i GID użytkownika, który uruchomił dany proces

## Proces rodzic

W momencie uruchomienia procesu, tworzony jest duplikat tego procesu. Ten nowy proces jest nazywany potomkiem, a proces który go wywołał jest nazywany rodzicem. Proces potomek zastępuje kopię kodu stworzoną przez proces rodzica kodem, który powinien uruchamiać proces potomek.
Podczas wykonywania polecenia, powłoka czeka na zakończenie działania procesu potomka.
Po jego zakończeniu, proces rodzic kończy działanie procesu potomka i wyświetlany jest znak zachęty informujący o gotowości do uruchomienia kolejnego polecenia.

## Wyświetlanie informacji o procesach i PID’ach

Polecenie ps (ang. *process status*) wyświetla procesy uruchomione w systemie. Dla każdego procesu ps wyświetla identyfikator procesu (PID), identyfikator terminalu (TTY), całkowity czas wykonywania (TIME) i nazwę polecenia (CMD).

Składnia:

```
ps –options
```

Listę wszystkich dostępnych opcji możemy uzyskać za pomocą polecenia: man ps

## Wyświetlanie pełnej listy wszystkich procesów

Poniższy przykład przedstawia wyświetlenie pełnej listy wszystkich procesów:

```
$ ps -ef | more
UID   PID   PPID       C    STIME      TTY   TIME CMD
root  0     0          0    16:46:41   ?     0:01 sched
root  1     0          0    16:46:44   ?     0:40 /etc/init -
root  2     0          0    16:46:44   ?     0:00 pageout
root  3     0          0    16:46:44   ?     4:33 fsflush
```

## Nagłówki kolumn dla polecenia ps -ef

- UID Nazwa użytkownika będącego właścicielem procesu
- PID Unikalny numer identyfikujący proces
- PPID Numer identyfikacyjny procesu rodzica dla procesu
- C procent wykorzystania CPU dla procesu
- STIME Czas rozpoczęcia procesu (hh:mm:ss)
- TTY Terminal z którego uruchomiono proces (terminal kontroli dla demonów systemowych ma oznaczenie [?])
- TIME Całkowity czas wykonywania procesu
- CMD Nazwa polecenia

## Wyszukiwanie konkretnego procesu

W celu szybkiego wyszukania konkretnego procesu, można przesłać wynik uruchomienia polecenia ps do polecenia grep w celu znalezienia przy jego pomocy konkretnej nazwy polecenia. Na przykład aby znaleźć wszystkie aktywne procesy związane z drukowaniem należy uruchomić następujące polecenie:

```
$ ps -e | grep lp

217 ? 0:00 lpsched
$
```

Uwaga: lpsched jest demonem odpowiedzialnym za kontrolę usług drukowania w systemie.

## Polecenie pgrep

Polecenie pgrep daje możliwość wydajnego wyszukiwania procesów po nazwie. Domyślnie pgrep wyświetla PID każdego procesu, który pasuje do podanego kryterium wyszukiwania.

```
pgrep -options pat tern
```

Opcje:

Możliwe jest użycie następujących opcji z poleceniem pgrep:

- -x Wyświetla tylko numery PID, które pasują dokładnie do kryterium wyszukiwania.
- -n Wyświetla tylko najnowszy (utworzony ostatnio) PID, który pasuje do kryterium wyszukiwania. (Wyświetla wszystkie ID procesów zawierających kryterium wyszukiwania).
- -U Wyświetla tylko identyfikatory PID procesów należących do konkretnego użytkownika. (Używa albo nazwy użytkownika albo identyfikatora UID).
- -l Wyświetla nazwę procesu wraz z jego numerem PID.

Uwaga: Polecenie pgrep posiada kilkanaście innych użytecznych opcji. Zaleca się zapoznanie z dokumentacją man dotyczącą tego polecenia, która zawiera pełną listę jego opcji.

W poniższych przykładach, użycie opcji –l wyświetla zarówno nazwy procesów jak i ich numery PID:

```
$ pgrep -l lp
217 lpsched

$ pgrep -l mail
230 sendmail
13453 dtmail
```

W następnym przykładzie, połączenie opcji –l i –x wyświetla tylko te procesy, które dokładnie pasują do podanego kryterium wyszukiwania:

```
$ pgrep -lx dtmail
12047 dtmail
```

## Wysyłanie sygnałów do procesów

Sygnały są używane do kontrolowania procesów uruchomionych w systemie. Sygnały są wysyłane do procesów w celu poinformowania procesu o wystąpieniu zdarzenia i wymuszenia odpowiedzi od procesu.
Na przykład naciśnięcie Control-C w celu zatrzymania polecenia wysyła sygnał przerwania do procesu na który proces musi odpowiedzieć zakończeniem działania.
Sygnał jest prostym komunikatem zawierającym numer sygnału, który jest przesyłany do procesu. W systemie Linux występuje określona liczba numerów sygnałów. Każdy sygnał posiada unikalny numer, nazwę i jest powiązany z określoną akcją.
Informacje szczegółowe dotyczące obsługi sygnałów możemy znaleźć w dokumentacji man:

```
$ man signal
```

```
$ man kill
```

## Wybrane sygnały, ich numer i nazwy

- 1, SIGHUP – Sygnał odwieszenia powoduje rozłączenie lini telefonicznej lub połączenia terminalu.
- 2, SIGINT – Sygnał przerwania generowany z klawiatury – zazwyczaj przez Control-C
- 9, SIGKILL – Sygnał używany do zabicia procesu. Nie może zostać zignorowany.
- 15, SIGTERM – Sygnał do zakończenia procesu w sposób uporządkowany. Niektóre procesy mogą zignorować ten sygnał.

## Kończenie działania procesu

Poniższa sekcja przedstawia metody używania polecenia kill.

## Polecenie kill

Polecenie kill używane jest do wysyłania sygnału do jednego lub więcej uruchomionych procesów. Polecenie to jest często używane do kończenia działania procesów.

Uwaga: Zwykły użytkownik może użyć polecenie kill tylko do swoich procesów. Użytkownik root może wywołać polecenie kill dla niemal wszystkich procesów.

Składnia:

```
kill signal PID ...
```

## Zakończenie działania procesu

Przed zatrzymaniem działania procesu należy poznać jego numer PID. W celu określenia numeru PID procesu używamy polecenia ps lub pgrep.

W celu zabicia procesu uruchamiamy polecenie:

```
$ kill PID
```

Przykład:

```
$ pgrep -l mail
215 sendmail
12047 dtmail

$ kill 12047
```

W celu zabicia więcej niż jednego procesu równocześnie uruchamiamy polecenie:

```
$ kill PID PID PID PID
```

Użycie polecenia kill bez określenia sygnału w linii poleceń wysyła do procesu sygnał o domyślnej wartości 15. Sygnał ten zazwyczaj zmusza proces do zakończenia działania.
Niektóre procesy ignorują sygnał o numerze 15. Mogą być to procesy oczekujące na dostęp do zasobów, na przykład proces oczekujący na napęd DVD kończący operację przesyłania danych w celu kontynuacji działania.
Procesy, które nie reagują na sygnał 15 mogą być zmuszone do zakończenia działania poprzez wysłanie do nich sygnału o numerze 9.

Przykład:

```
$ kill -9 PID
```
lub
```
$ kill -KILL PID
```

Uwaga: Użycie polecenia kill -9 lub polecenia kill –KILL jest ostatnim środkiem do zakończenia działania procesu. Użycie polecenia kill -9 lub polecenia kill –KILL na procesie, który kontroluje aplikację bazodanową lub program, który uaktualnia pliki może mieć niepożądane skutki uboczne. Proces tego typu jest gwałtownie przerywany bez możliwości zakończenia operacji w sposób uporządkowany co może prowadzić do utracenia pewnej ilości danych lub niepoprawnego zapisania struktury pliku na dysku.

## Polecenie pkill

Polecenie pkill służy do zatrzymywania działania procesów domyślnie wysyłając do nich sygnał o numerze 15. Polecenie to może zatrzymać działanie procesu poprzez podanie nazwy procesu.

Składnia:

```
pkill [-options] pat tern
```

Opcje polecenia pkill są takie same jak dla polecenia pgrep.

W celu zakończenia procesu sesji poczty poprzez podanie jej nazwy wywołujemy:

```
$ pgrep -l mail
215 sendmail
470 dtmail
$ pkill dtmail
```

W celu zakończenia procesu sesji dtmail poprzez podanie jej numeru PID używamy polecenia kill:

```
$ kill 470
```

## *Zarządzanie zadaniami (jobs)*

Zadanie jest procesem zarządzanym przez terminal i zawierającym ID procesu. Każde zadanie ma przypisany identyfikator nadany przez powłokę. Obsługa wielu zadań przez powłokę jest nazywana kontrolą zadań.
Powłoka umożliwia równoczesne uruchomienie wielu zadań. Otwieranie aplikacji, wysyłanie żądania drukowania lub uruchomienie polecenia ls na katalogu są przykładami zadań.
Kiedy zadanie jest wykonywane w środowisku okienkowym, jego działanie jest widoczne w linii poleceń i jest związane z tym konkretnym oknem aż do momentu jej ukończenia.
Jednakże, możliwe jest uruchamianie zadań przez powłokę działających w tle i uwolnienie okna w celu uruchomienia w nim kolejnego polecenia. Możliwe jest kontrolowanie działania zadań poprzez ich numery. Zadania można kontrolować za pomocą następujących poleceń:

- jobs - Wyświetla obecnie uruchomione zadania
- bg %n Umieszcza określone zadanie w tle (n jest numerem zadania)
- fg %n Przywraca określone zadanie z tła (n jest numerem zadania)
- ^Z Zatrzymuje bieżące zadanie
- stop %n Zatrzymuje określone zadanie działające w tle (n jest numerem zadania)

Uwaga: Kontrolowanie zadania przy użyciu tych poleceń jest możliwe tylko z poziomu okna w którym zostały uruchomione zadania.

W celu uruchomienia zadania w tle, wpisz polecenie, które ma być uruchomione, a po nim znak ampersand (&).

Przykład uruchomienia polecenie find w tle. Znajduje ono wszystkie pliki o nazwie core w bieżącym katalogu. Następnie zapisuje ono pełną ścieżkę każdego znalezionego pliku zawierającego nazwę core do pliku o nazwie list.

```
$ find . -name core > list &
[1] 3028
$
```

Powłoka zwraca numer ID zadania w nawiasach kwadratowych i numer PID polecenia. Numer ID zadania daje nam możliwość jego kontroli. Numer PID jest używany głównie przez kernel do zarządzania zadaniami.

Po naciśnięciu klawisza Enter w oknie wyświetlany jest komunikat informujący o zakończeniu działania zadania:

```
[1] + Done find . -name core > list &
$
```

Polecenie jobs pozwala na wyświetlenie obecnie działających zadań:

```
$ jobs
[1] + Running find . -name core > list &
```

Polecenia fg służy do przywrócenia zadania z tła:

```
$ fg %1
find . -name core > list
```

Uwaga: To polecenie blokuje okno aż do momentu zakończenia zadania, zatrzymania zadania lub zatrzymania i przeniesienia zadania z powrotem w tło.

W celu przeniesienia zadania w tło, możemy go zatrzymać przy pomocy skrótu klawiszowego ^Z, a następnie użyć polecenia bg:

```
$ find . -name core > list
^Z
[1] + Stopped(SIGTSTP) find . -name core > list &
$ jobs
[1] + Stopped(SIGTSTP) find . -name core > list &
$ bg %1
[1] find . -name core > list &
```

Uwaga: Umieszczenie zatrzymanego zadania w tle lub we froncie okna powoduje jego ponowne uruchomienie.

W celu zatrzymania zadania w tle, można użyć numeru tego zadania jako argumentu dla polecenia stop lub kill:

```
$ stop %1
[1] + Stopped (SIGSTOP) script1 &
$

$ kill %1
[1] + Stopped (SIGSTOP) script1 &
```

## Zadania:

1. Uruchom następujące polecenie w tle: sleep 500 &
2. Używając polecenia jobs, znajdź numer zadania dla polecenia wywołanego w poprzednim punkcie.
3. Przenieś to zadnie z tła, a następnie z powrotem umieść w tle.
4. Zatrzymaj zadanie wykonujące polecenie sleep przy użyciu polecenia kill.
5. Użyj następujących poleceń ps w celu wyświetlenia procesów obecnie uruchomionych w systemie. Jakiej informacji dostarcza każde z tych poleceń ?

   $ ps
   $ ps -f
   $ps -e
   $ps -ef

6. W oknie terminala uruchom polecenie ps –ef. Zidentyfikuj numer PID tego polecenia.
7. W osobnym oknie terminalu uruchom polecenie cat –v /dev/zero.

   Uwaga: To polecenie jest używane do tworzenia ciągle działającego procesu dla potrzeb demonstracyjnych. Informacja na temat pliku /dev/zero znajduje się w dokumentacji i można ją wyświetlić przy pomocy polecenia man zero.

8. Otwórz kolejne okno terminalu i przy pomocy polecenia ps zidentyfikuj PID polecenia cat.
9. W tym samym oknie terminalu zatrzymaj polecenie cat przy pomocy polecenia kill podając numer PID polecenia cat.
10. W tym samym oknie terminalu wpisz polecenie tty w celu identyfikacji tego okna. Nazwa zostaje wyświetlona jako /dev/pts/#, gdzie # reprezentuje numer okna np. /dev/pts/4.
11. Przejdź do pierwszego okna terminalu i użyj polecenia pgrep do identyfikacji numeru PID związanego z drugim oknem terminalu.
12. W bieżącym oknie terminalu użyj polecenia kill do zatrzymania działania drugiego okna terminala. kill PID#

    Czy polecenie zatrzymało okno ?
13. Użyj polecenia kill z opcją -9 do zatrzymania działania drugiego okna terminalu.
    kill -9 PID#
    Czy polecenie zatrzymało okno ?
14. Uruchom powłokę Korna w pozostałym oknie przy pomocy polecenia ksh.
15. Użyj polecenia kill i zidentyfikuj sygnały, które są przez to polecenie wysyłane po wywołaniu go z określonymi opcjami:

    kill -l -9
    Sygnał:
    Kill –l -15
    Sygnał:

16. Wyjdź z powłoki Korna przy użyciu polecenia exit.
17. Utwórz skrypt bash, który uruchamia długo trwającą komendę, na przykład sleep 1000 &.
18. Użyj ps lub pgrep, aby znaleźć PID tej komendy.
19. Użyj kill -STOP, aby wstrzymać wykonanie procesu.

20. Użyj kill -CONT, aby wznowić wykonanie tego procesu.
21. Napisz skrypt bash, który monitoruje wybrany proces, i jeśli przekroczy on wykorzystanie CPU powyżej 50%, to monitorowany proces powinien zostać zatrzymany (podpowiedź: przydatne mogą być następujące polecenia: ps, watch, top)
22. Napisz skrypt bash, który dla podanego zakresu wykorzystania procesora w procentach, wyświetla co sekundę informacje, które procesy spełniają podane kryteria.

    Przykładowa sesja:

```
$ ./cpu_usage.sh 20% 50%
mc 25%
rsync 34%
md5sum 42%
```