Procesy w systemie UNIX

Proces to instancja wykonywanego programu. Systemy operacyjne mogą obsługiwać wiele procesów naraz, każdemu z nich przydzielając numer identyfikacyjny – *PID* (ang. *process identifier*) oraz odpowiednie zasoby, takie jak pamięć czy czas procesora.

Poniżej podajemy przykładowy proces, który powinien trwać na tyle długo, aby miało sens mierzenie mu czasu komendą `time` – próbuje przeglądać zawartość katalogu głównego i rekurencyjnie zawartość jego podkatalogów, ale nie będzie nic wypisywał na ekran konsoli (symbol `&>` oznacza jednoczesne przekierowanie standardowego wyjścia danych `>` i standardowego wyjścia błędów `2>` do podanego pliku – w tym wypadku specjalnego pliku, który ignoruje wszystkie dane zapisane do niego):

```
time ls -R / &> /dev/null
⟶
⟶real    0m2.594s
⟶user    0m1.234s
⟶sys     0m1.336s
```

Czas `real` to faktyczny czas wykonania programu, czas `user` to czas poświęcony na wykonywaniu samych instrukcji programu bez ingerencji w polecenia systemu operacyjnego, natomiast czas `sys`, to czas poświęcony na wykonywanie instrukcji w jądrze systemu operacyjnego.

Trochę zmodyfikujemy nasze ostatnie polecenie, aby zapisywało swój rezultat do normalnego pliku: `ls -R / &> ~/katalogi`
Z poziomu konsoli możemy przerwać działanie tego programu przy pomocy kombinacji Ctrl-C. Po wciśnięciu klawiszy wyemitowany zostanie sygnał `SIGINT`, który powinien spowodować wymienione przerwanie, o ile nie zostanie wcześniej przechwycony. Za inną kombinacją, Ctrl-Z, kryje się sygnał `SIGTSTP`, który wstrzymuje działanie programu. Program wstrzymany można ponownie uruchomić od momentu, w którym nastąpiło wstrzymanie. Jeśli użyliśmy tej ostatniej kombinacji, to zobaczymy status zadania – zatrzymane.

Przy pomocy polecenia `jobs` możemy zobaczyć stan wszystkich zadań w aktualnej sesji. Możemy uruchomić i zatrzymać w podany sposób kolejny proces: `sleep 1000` ("nic nie rób przez 1000 sekund"), co pozwoli nam zobaczyć dwa numerowane wpisy na liście zadań (pytanie: jaki będzie rezultat wykonania `time sleep 5`?). Jeśli chcemy zakończyć pierwszy z tych procesów, to możemy zrobić to przy pomocy polecenia `kill %1` odwołującego się do numeru na liście zadań poprzedzonego znakiem `%`.

Alternatywnie, jako argument polecenia `kill` możemy użyć numeru PID. Uzyskamy go dzięki poleceniu `ps`, które wypisuje listę naszych procesów związanych z konsolą. Inne przykłady wywołań polecenia `ps`:

`ps x`	wszystkie nasze procesy, także niezwiązane z konsolą;
`ps ux`	jak wyżej, wraz dodatkowymi informacjami, np. o właścicielu procesu i czasie startu;
`ps ax`	wszystkie procesy wszystkich osób na serwerze;
`ps axo pid,user,tty,command`	to co wyżej, ale w czterech następujących kolumnach: PID, nazwa użytkownika, konsola, treść komendy.

Teraz możemy np. usunąć proces o numerze PID 4493 poprzez komendę `kill 4493`. Polecenie `kill` standardowo wysyła sygnał `SIGTERM`, który jednak, podobnie jak `SIGINT`, może zostać przechwycony przez program. W przypadkach krytycznych należy zatem używać opcji `-9` (np. `kill -9 4493`), powodującej wysłanie sygnału `SIGKILL`, który działa jak `SIGTERM`, ale nie da się go przechwycić. Jeżeli naszym celem jest zakończenie od razu wszystkich procesów związanych z konkretną komendą, to możemy użyć polecenia `killall`, np. `killall sleep`.

```
#include<stdio.h>
#include<signal.h>

void sighandler(int sig) {
  printf("Caught signal %d. Avoiding ^C.\n", sig);
}

int main() {
  signal(SIGINT, &sighandler);
  while(1);
  return 0;
}
```

Istniejące zadanie (np. o numerze 2 na liście `jobs`) można zmusić do pracy w tle przy pomocy polecenia `bg %2`. Jeśli jednak od razu chcemy wykonać dane zadanie w tle, to wystarczy na jego końcu wstawić znak `&`. **Uwaga:** procesy działające w tle będą działały nadal po zakończeniu sesji! Przykład polecenia:

```
ls -l / &> ~/katalogi2 &
⟶[3] 5988
```

Od razu otrzymaliśmy w ten sposób numer zadania i numer PID. Przyjmijmy, że po wywołaniu komendy `jobs` otrzymamy:

```
⟶[1]+  Stopped                 ls -R / >&katalogi
⟶[2]   Running                 sleep 1000 &
⟶[3]-  Done                    ls -l / >&katalogi2
```

Zadanie o numerze 3 zakończyło się, więc przy kolejnym wywołaniu `jobs` już nie pojawi się na liście. Drugie zadanie działa w tle, natomiast pierwsze wciąż jest zatrzymane – możemy je przenieść do pracy na pierwszym planie przy pomocy polecenia `fg %1`.

Zarządzać procesami można przy użyciu poleceń `top` lub `htop`. Aby uzyskać pomoc w trakcie działania któregoś z tych programów, wciskamy jeden z klawiszy `h` lub `?`. Wyjście odbywa się przy pomocy `q`. `htop` jest bardziej przejrzysty i wygodny w obsłudze niż `top`, ale ten drugi z kolei oferuje możliwość zapisu raportu do pliku. Poniższe akapity są poświęcone programowi `top`.

W celu zobaczenia procesów należących wyłącznie do użytkownika `user1` możemy użyć komendy: `top -u user1`. Alternatywnie, w trakcie działania programu `top` wciskamy klawisz `u`, wpisujemy `user1` i akceptujemy. Jeżeli nie napiszemy żadnej nazwy, to zobaczymy procesy wszystkich użytkowników. Kończymy działanie danego procesu wciskając `k` i podając jego PID. Przy pomocy polecenia `top -p pid1, pid2, ...` (np. `top -p 1, 3`) możemy śledzić wyłącznie procesy o podanych PIDach (w przykładzie: `1` i `3`). Klawisze `z` i `b` wpływają na oznaczenie aktywnych procesów.

Procesy możemy sortować względem wybranej kolumny. Domyślnie posortowane są po procentowym zużyciu procesora (klawisz `P`). Jeśli chcemy ułożyć je po zużyciu pamięci, to używamy klawisza `M`. W przypadku innego sortowania używamy `O` lub `F`, natomiast `R` odwraca sortowanie. Możemy także decydować o wyborze kolumn do wyświetlenia (`f`) i ich kolejności (`o`). Klawisz `c` (jak i opcja `-c` przy uruchomieniu) sprawia, że możemy przełączać się między mniej i bardziej szczegółowym widokiem komend związanych z poszczególnymi procesami. Przy pomocy `l`, `t` i `m` można włączać lub wyłączać podsumowania nad listą procesów.

Domyślny czas, po którym odświeżana jest lista procesów wynosi 3 sekundy. Aby go zmodyfikować, używamy albo komendy `top -d liczba_sekund`, albo klawiszy `d` lub `s` w trakcie działania polecenia `top`. Jeżeli chcemy natychmiast poznać aktualny stan listy procesów, to używamy enter bądź spacji.

W celu zapisania aktualnego stanu procesów do pliku używamy polecenia:
`top -bn 1 > plik`

Wykonywać różne zadania w tym samym czasie z przełączaniem się na pierwszy plan i tło można także przy użyciu polecenia `screen`. W praktyce ten program tworzy w osobnych procesach wirtualne terminale, między którymi można się łatwo przemieszczać. Podobnie jak w przypadku procesów działających w tle, wirtualne terminale będą istnieć także po zakończeniu sesji.

Taki wirtualny terminal można zainicjalizować, wpisując `screen komenda` lub po prostu `screen`, a potem wpisując komendę. W tym pierwszym przypadku jednak wirtualny terminal zniknie wraz z zakończenie procesu. Przykładowo:
`screen sleep 2000`

Aby wrócić do głównego terminala, należy użyć kombinacji `Ctrl-a d`. W ten sposób można opuścić wirtualny terminal, jednocześnie nie usuwając go. Przekonamy się o tym, wpisując komendę `screen -ls`, która wylistuje aktualne sesje `screen`a. Wcześniej jednak

wywołamy `screen -S nazwa`, która utworzy nam wirtualny terminal o ustalonej nazwie i opuścimy go. Zatem:

```
screen -ls
⟶There are screens on:
⟶        5506.pts-0.tcs  (Detached)
⟶        8367.nazwa      (Detached)
⟶2 Sockets in /tmp/uscreens/S-kosinski.
```

Poszczególne `screeny` wypisane są w postaci: `PID.nazwa_terminala[.host]`. Jeżeli chcemy wejść do jednego z tych wirtualnych terminali (powiedzmy: `5506.pts-0.tcs`), to wpisujemy jedno z następujących poleceń (ostatnie dwa z nich zadziałają, o ile nazwa wraz z ewentualnym hostem jest unikalna):

```
screen -r 5506.pts-0.tcs
screen -r 5506
screen -r pts-0
screen -r pts-0.tcs
```

Ciekawostka: możemy opuścić aktualny wirtualny terminal z innego. . . "fizycznego" terminala, wpisując `screen -d identyfikator`, gdzie `identyfikator` jest taki, jak w przypadku opcji `-r`.

Wewnątrz `screen`a tworzymy kolejne wirtualne podterminale przy pomocy kombinacji `Ctrl-a c`. Możemy się między nimi przemieszczać przy pomocy kombinacji:

`Ctrl-a n`	następny podterminal,
`Ctrl-a p`	poprzedni podterminal,
`Ctrl-a 0-9`	skok do podterminala o podanym numerze (od 0 do 9),
`Ctrl-a '`	skok do podterminala o numerze podanym po `'`,
`Ctrl-a "`	przejście do menu wyboru podterminala,
`Ctrl-a Ctrl-a`	przemieszczenie się między 2 ostatnio używanymi podterminalami.

Ogólnie wewnątrz `screen`a używamy poleceń, które zaczynają się od `Ctrl-a` (patrz pomoc – `Ctrl-a ?`). W celu zamknięcia podterminala używamy `Ctrl-a K`, ewentualnie komendy `exit`. Po zamknięciu wszystkich podterminali w danym `screen`ie, ten ostatni kończy działanie.