Pojęcie zadania w systemie operacyjnym ma wiele znaczeń. O które chodzi bardzo często wynika dopiero z kontekstu. W oryginalnej, angielskiej terminologii przyjętej w zakresie systemów operacyjnych używa się trzech różnych słów:: *process*, *job*, *task*. Ten wykład poświęcony jest zadaniom w sensie tych dwóch ostatnich.

Poprzez **zadanie** w sensie **job** rozumiemy proces związany z konkretnym terminalem, na którym został uruchomiony. Proces jest pojęciem szerszym, każde zadanie to proces, ale nie na odwrót. Jest w systemie wiele procesów, nie związanych z żadnym terminalem. Obecnie, gdy mamy systemy okienkowe i możemy otworzyć sobie wiele okienek z terminalem, pojęcie zadania ma mniejsze znaczenie niż kiedyś, gdy użytkownik po zalogowaniu miał dostęp wyłącznie do jednego terminala. Nader często było tak, że użytkownik uruchamiał program analizujący jakieś dane, co trwało znacznie dłużej niż przerwa na kawę, i chciał sprawdzić pocztę w trakcie działania tego programu. Należało wstrzymać proces zajmujący terminal , albo przenieść go w tło i wtedy można było uruchomić inny program. Stąd mamy w Unixie narzędzia do obsługi procesów w konkretnym terminalu.

Gdy uruchamiamy program w terminalu to trzy standardowe strumienie : wejścia, wyjścia i błędów kojarzone są z naszym procesem. Dopóki nasz proces się nie zakończy, nie możemy korzystać z terminala. Mówimy, że taki program działa **na pierwszym planie**. Dopisując znak `&` na końcu polecenia uruchamiającego program spowodujemy, że nie przekażemy standardowego wejścia terminala naszemu procesowi. Program się uruchomi i o ile nie pobiera danych ze standardowego wejścia, będzie działał poprawnie **w tle**. Formalnie oznacza to tyle, że nie ma on dojścia do standardowego wejścia terminala. Dla nas oznacza to, że nasz program działa, a my dalej możemy używać klawiatury do wprowadzania poleceń w shellu terminala.

```
sirius$ nedit &
[1] 3000
sirius$ gcalctool &
[2] 3002
sirius$ jobs
[1]-  Running                 nedit &
[2]+  Running                 gcalctool &
sirius$ pgrep gcalctool
3002
```

Zadania, podobnie jak procesy, są ponumerowane. Ich identyfikatory to liczby naturalne od 1. Gdy chcemy odwołać się do zadania podajemy tę liczbę poprzedzoną znakiem procent `%`. W powyższym przykładzie, mamy dwa zadania: `nedit` i `gcalctool` uruchomione w tle. Świadczy o tym znak `&` na końcu listy zadań wyświetlanej poleceniem `jobs`. Numerek w nawiasach kwadratowych oznacza numer zadania natomiast liczby 3000 i 3002, które pojawiły się w momencie uruchamiania programów to identyfikatory PID odpowiednich procesów. Sprawdziliśmy to poleceniem `pgrep`.

Każde zadanie może byż w jednym z trzech następujących stanów:

- uruchomione na pierwszym planie (foreground),
- uruchomione w tle (background),
- zatrzymane.

Ponieważ jest tylko jeden strumień standardowego wejścia, więc tylko jedno zadania może działać na pierwszym planie. Pozostałe strumienie są współdzielone, to znaczy, gdy zadania w tle zaczynają pisać na standardowe wyjście to zobaczymy wyniki jednego zadania przeplatane z pozostałymi.

Wróćmy do historycznego problemu, o którym pisaliśmy na początku. Spróbujmy odpowiedzieć na pytanie jak wstrzymać działanie zadania nie przerywając go. Generalnie, należy wysłać do danego zadania sygnał `SIGSTOP`. Można to zrobić na trzy sposoby:

1. `kill -STOP %N`
2. `kill -STOP PID`
3. `Ctrl+Z`

gdzie `N` to numer zadania, a `PID` to identyfikator procesu. W przypadku, gdy zadanie jest pierwszoplanowe to rozwiązanie (1) odpada, bo nic nie możemy wprowadzić z klawiatury w danym terminalu. Rozwiązanie drugie wymaga otwarcia drugiego okienka terminala i wyszukanie PID procesu. Wtedy z tego, drugiego okienka możemy wpisać polecenie (2). Najprostsze rozwiązanie to wciśnięcie kombinacji `Ctrl+Z` nawet, gdy zadanie jest na pierwszym planie. Nie zadziała ono jednak z aplikacjami okienkowymi. W wielu edytorach `Ctrl+Z` to cofnięcie ostatniej operacji. Takich jednak nie było w opisywanej sytuacji z historii.

Gdy wszystkie zadania działają w tle, to swobodnie możemy korzystać z poleceń `jobs`, `fg` oraz `bg`. Nazwy są dość sugestywne. Działanie pierwszego już widzieliśmy. Z opcją -l dostajemy identyfikatory PID procesów:

```
sirius$ jobs -l
[1]  3457 Running      ping -s wp.pl 48 105 >/tmp/x &
[2]- 3459 Running      iostat -xn 3 >/tmp/y &
[3]+ 3478 Running      rsync -vruplt /data/mail /net/omega/backup/ >/tmp/z &
```

Znak `+` oznacza bieżące zadanie, ostatnio modyfikowane. Znak – to stoi przy zadaniu poprzednim. Zamiast identyfikatorów zadań w poleceniach można podawać `%+` albo `%-`. Tutaj mamy 3 zadania. Komentarz `Running` oznacza, że zadania jest wykonywane. Wszystkie zadania działają w tle. Poznajemy to po znaku `&` na końcu polecenia w ostatniej kolumnie. Możemy teraz przenieść zadanie 2 na pierwszy plan:

```
sirius$ fg %2
iostat -xn 3 >/tmp/y
^Z
[2]+  Stopped        iostat -xn 3 >/tmp/y
```

Wciśniecie `Ctrl+Z` spowodowało zatrzymanie zadania 2. Shell wyświetla nam o tym komunikat. Teraz lista zadań wygląda tak:

Przy zadaniu 2 pojawiło się `Stopped`. To oznacza, że wykonanie zadania zostało wstrzymane. Gdy któreś z zadań zakończy się to zobaczymy komentarz `Done`.

```
sirius$ jobs
[1]   Done          ping -s wp.pl 48 105 >/tmp/x &
[2]+  Stopped       iostat -xn 3 >/tmp/y &
[3]-  Running       rsync -vruplt /data/mail /net/omega/backup/ >/tmp/z &
```

Aby wznowić wykonanie zadania mamy kilka możliwości. Możemy je przenieść na pierwszy plan korzystając z fg, albo wznowić w tle poleceniem bg. Możemy też wysłać do niego sygnał `SIGCONT` poleceniem `kill`. To znaczy odpowiednio:

1. `fg %2`
2. `bg %2`
3. `kill -CONT %2`

Wysłanie sygnału kontynuacji wznawia zadanie w tle.