

Systemy operacyjne

(slajdy uzupełniające)

Wykład 3: Terminal i zarządzanie zadaniami

Czym są zadania powłoki?

- Zadanie to grupa procesów uruchomiona w wyniku wpisania polecenia.
- Polecenie może łączyć procesy potokiem (znak pałki) i będą one wykonywane współbieżnie.
`grep ded /usr/share/dict/words | wc -l`
- Polecenia oddzielone znakiem średnika są uruchamiane jako ciąg, jedno po drugim:
`echo -n "hello "; sleep 2; echo "world"`
- Kod wyjścia ostatniego procesu w potoku lub ciągu będzie kodem wyjścia zadania:
`echo $?`

Co wiemy o zarządzaniu zadaniami w powłoce?

- Jeśli po poleceniu występuje znak `'&'` to zadanie uruchomi się w grupie drugoplanowej.
- Wszystkie zadania drugoplanowe można wypisać poleceniem powłoki **jobs**.
- Wbudowane polecenie powłoki (np. **kill**) rozpoznają ciąg znaków `'%n'` jako zadanie o identyfikatorze **n**.
- Zadanie wstrzymane znakiem **CTRL+Z** można kontynuować w tle (**bg**) lub przenieść do grupy pierwszoplanowej (**fg**).

Co już wiemy o pliku terminala?

- Urządzenie znakowe do odczytu ([read](#)) i zapisu ([write](#)).
- Obsługiwane przez sterownik znajdujący się w jądrze.
- Buforuje wejście od użytkownika umożliwiając edycję linii.
- Po naciśnięciu klawisza **ENTER** wysyła linię do procesu.
- Do grupy pierwszoplanowej grupy procesów dany znak wysyła sygnał: **CTRL+C** → **SIGINT**, **CTRL+S** → **SIGSTOP**, **CTRL+Q** → **SIGCONT**, **CTRL+** → **SIGQUIT**
- ... a do grupy drugoplanowej: **CTRL+Z** → **SIGTSTP** (czyli *terminal stop*)
- Znak **CTRL+D** wysyła znacznik końca pliku, ale samego pliku nie zamyka.

Właściwości urządzenia terminala

```
# stty -a
```

```
speed 38400 baud; rows 32; columns 122; line = 0;
```

```
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = M-^?;
```

```
eol2 = M-^?; swtch = <undef>; start = ^Q; stop = ^S; susp = ^Z;
```

```
rprnt = ^R; werase = ^W; lnext = ^V; discard = ^O; min = 1; time = 0;
```

```
-parenb -parodd -cmspar cs8 -hupcl -cstopb cread -clocal -crtscts
```

```
-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl
```

```
ixon -ixoff -iuclic ixany imaxbel iutf8 opost -olcuc -ocrnl onlcr
```

```
-onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0 vt0 ff0 isig icanon
```

```
iexten echo echoe echok -echonl -noflsh -xcase -tostop -echoprt
```

```
echoctl echoke -flusho -extproc
```

Znaki oznaczone na **zielono** służą do edycji linii.

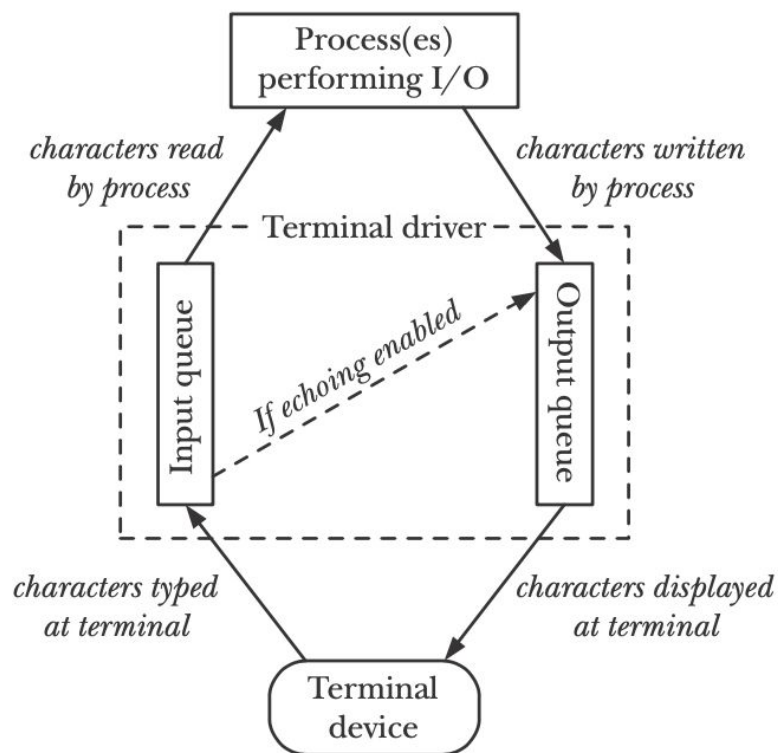
Domyślnie terminal jest w trybie kanonicznym

Tj. udostępnia edycję linii,
interpretuje znaki specjalne
i wypisuje na ekran znaki
wpisane przez użytkownika.

Kiedy użytkownik ma wpisać hasło
aplikacja musi wyłączyć **echo** w
terminalu →

`tcgetattr(3)`, `tcsetattr(3)`

Szczegóły w [termios\(4\)](#).



Interakcja zadań drugoplanowych z terminalem

Zadanie drugoplanowe, które

- czyta z terminala, to dostanie **SIGTTIN** (`'cat - &'`), a jeśli je ignoruje to **read** zwróci błąd,
- pisze do terminala, a ten ma ustawioną flagę **tostop**, to dostanie **SIGTTOU** (`'stty tostop; echo so20 &'`)

Domyślną akcją dla obydwu sygnałów jest zatrzymanie procesu tak, jakby dostały **SIGSTOP**.

Terminal sterujący a sesja

Sesja to pewna liczba grup procesów podłączonych do tego samego terminala sterującego.

Terminal sterujący może należeć do co najwyżej jednej sesji – można wyświetlić poleceniem ‘tty’.

Liderem sesji jest z reguły proces powłoki. Żeby zostać liderem sesji należy utworzyć sesję [setsid](#) i ustalić plik terminala sterującego [tcsetsid](#).

Jeśli powłoka zostanie odłączona od terminala, to dostanie **SIGHUP** i zdecyduje co zrobić z procesami, którymi zarządza.

Przykładowa sesja (LPI, s. 701)

Wyświetla pid powłoki:

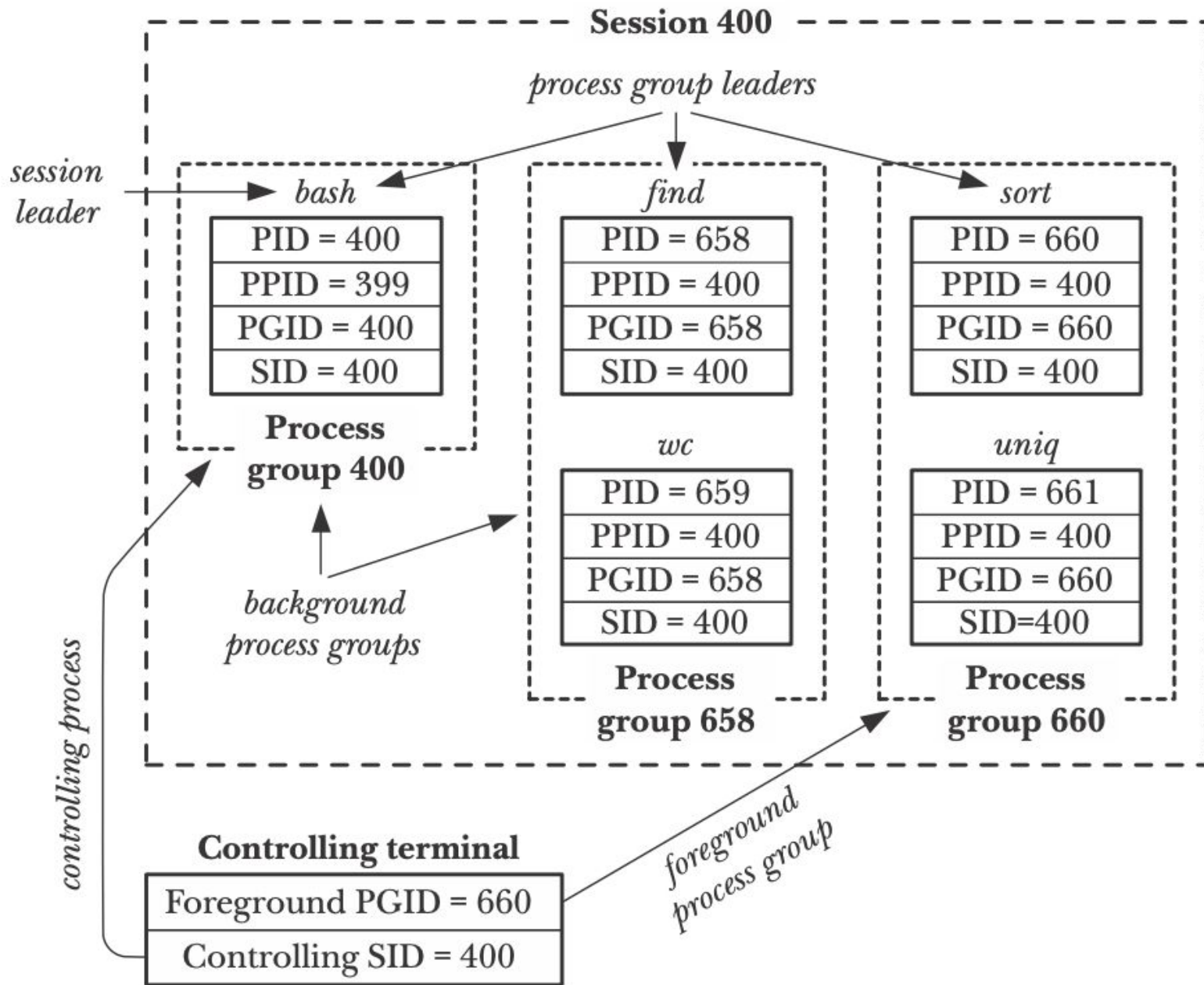
```
# echo $$  
400
```

Grupa drugoplanowa z dwóch procesów:

```
# find / 2> /dev/null | wc -l &  
[1] 659
```

Grupa pierwszoplanowa z dwóch procesów:

```
# sort < longlist | uniq -c
```



Powłoka a obsługa terminala

Powłoka musi wykrywać, które zadania zostały zatrzymane albo wznowione, by poprawnie utrzymywać listę zadań oraz ustalać pierwszoplanową grupę procesów → [tcsetpgrp](#).

Interakcja z użytkownikiem wymaga bycia w grupie pierwszoplanowej, inaczej dostaniemy **SIGTTIN**.

Okazuje się, że [waitpid\(2\)](#) służy nie tylko do oczekiwania na zakończenie procesu potomnego, ale ogólniej do obserwowania zmian stanu procesów potomnych → flagi **WSTOPPED** i **WCONTINUED**.

Pytania?