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 (polecenie **`bg`**) lub przenieść do grupy pierwszoplanowej (polecenie **`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 wierszami i umożliwia edycję wiersza.
- Po naciśnięciu klawisza **ENTER** wysyła wiersz do procesu.
- Do grupy pierwszoplanowej grupy procesów dany znak wysyła sygnał: **CTRL+C** → **SIGINT**, **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 -iuclc 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
```

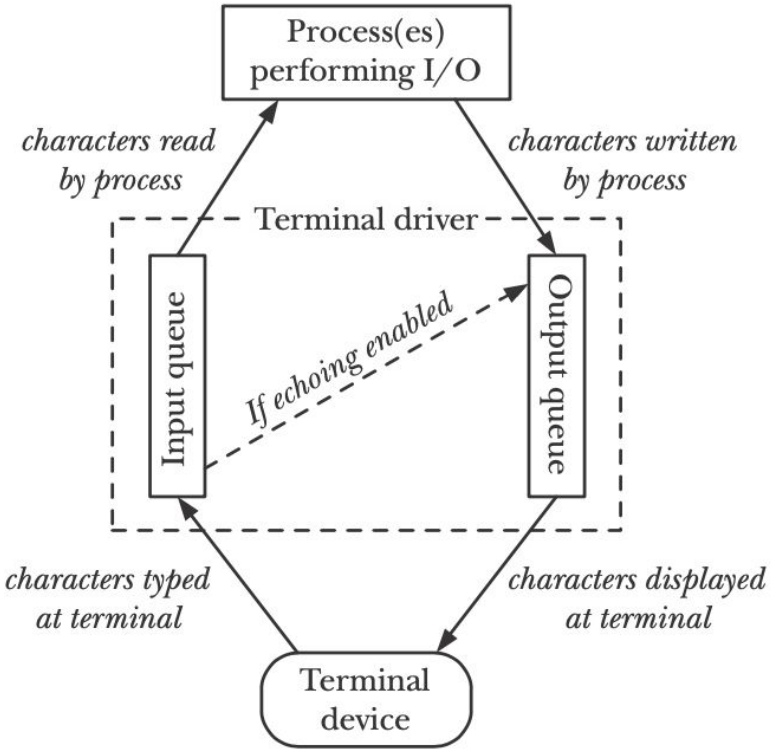
Znaki na **zielono** służą do edycji wiersza, na **fioletowo** do sterowania przepływem, **pogrubione** wysyłają sygnały.

Domyślnie terminal jest w trybie kanonicznym

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

Kiedy użytkownik ma wpisać hasło aplikacja musi wyłączyć flagę **`echo`** terminala → **`tcgetattr(3)`**, **`tcsetattr(3)`**.

Szczegóły można znaleźć 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 so21 &`**’)

Domyślną akcją dla obydwu w/w sygnałów jest zatrzymanie procesu. W odróżnieniu od **`SIGSTOP`** można te sygnały przechwycić, podobnie jak **`SIGTSTP`**!

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 bieżący poleceniem ‘`tty`’.

Liderem sesji jest z reguły proces powłoki. Żeby zostać liderem sesji należy utworzyć sesję `setsid` i ustalić terminal sterujący `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:

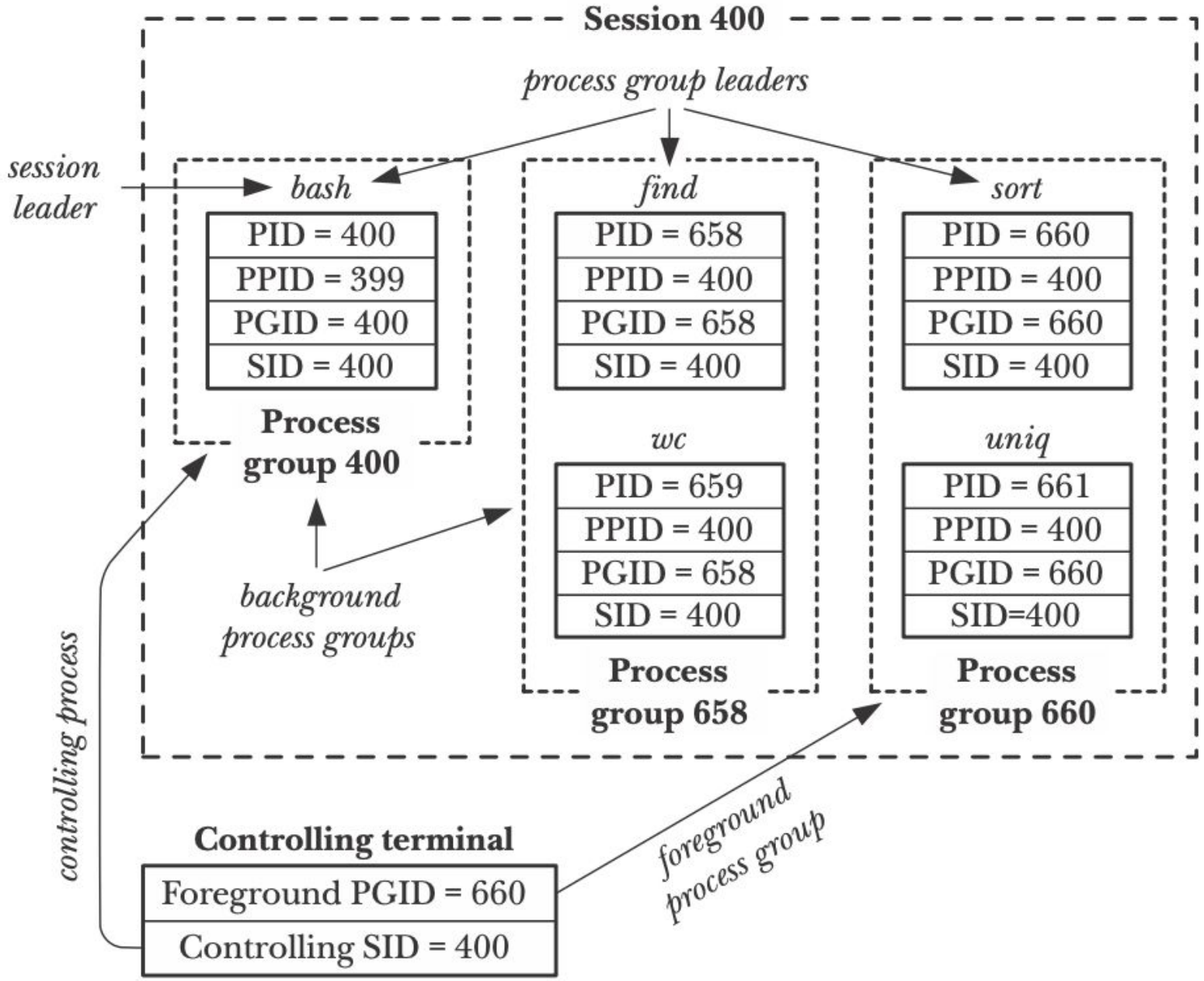
```
# echo $$
400
```

Grupa drugoplanowa z dwóch procesów:

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

Grupa pierwszoplanowa z dwóch procesów:

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


Session 400
process group leaders
session leader
bash
PID = 400
PPID = 399
PGID = 400
SID = 400
Process group 400
find
PID = 658
PPID = 400
PGID = 658
SID = 400
wc
PID = 659
PPID = 400
PGID = 658
SID = 400
Process group 658
sort
PID = 660
PPID = 400
PGID = 660
SID = 400
uniq
PID = 661
PPID = 400
PGID = 660
SID=400
Process group 660
background process groups
controlling process
Controlling terminal
Foreground PGID = 660
Controlling SID = 400
foreground process group

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`**.

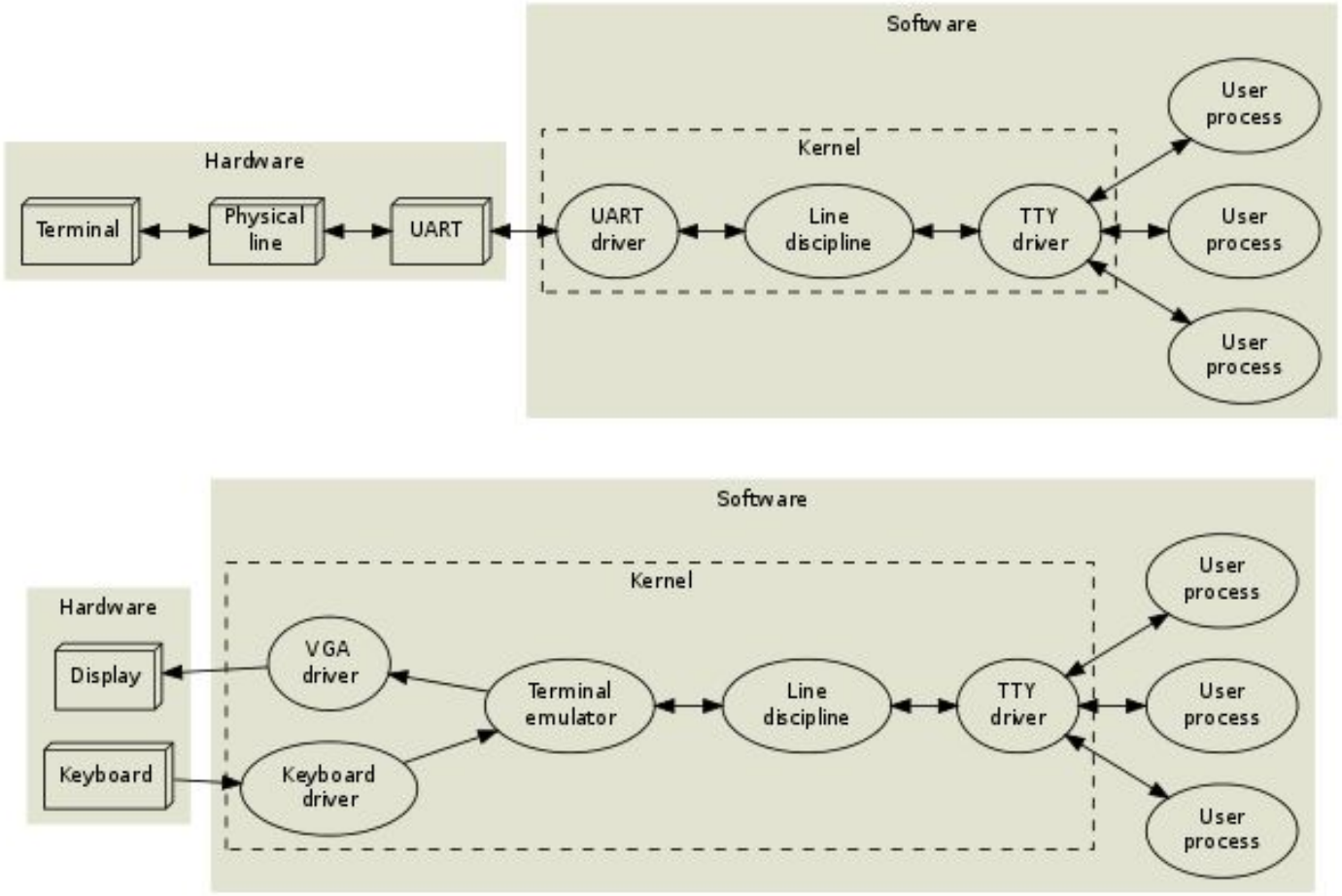
Emulatory terminala

Polecenie ‘`tty`’ w Waszym ulubionym emulatorze terminala (np. **`gnome-terminal`**, **`konsole`**, **`rxvt`**, **`xterm`**, itp.) zwróci ciąg znaków ‘`/dev/pts/N`’, gdzie `N` jest liczbą, zamiast oczekiwanego pliku terminala `/dev/ttyN`. Dlaczego?

Sterownik terminala w jądrze może użyć:

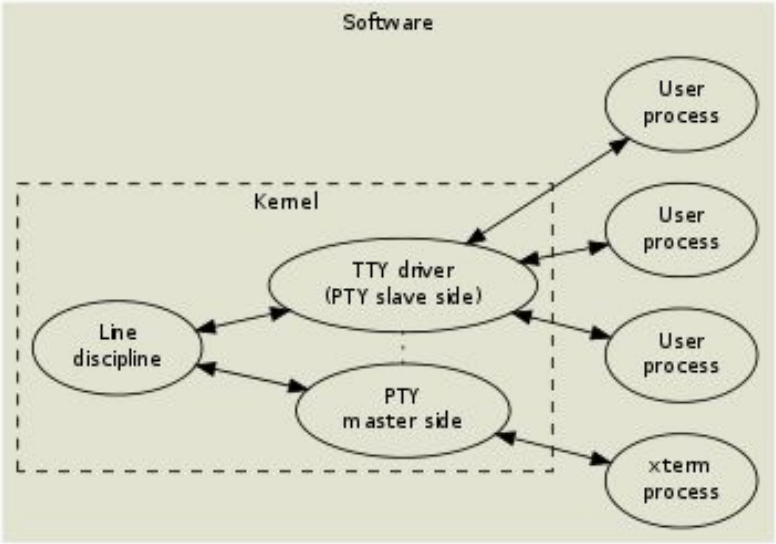
1. sterownika łącza szeregowego, żeby oddelegować zadanie pobierania znaków od użytkownika i wyświetlania zawartości terminala do fizycznego urządzenia terminala
2. użyć sterownika karty graficznej i klawiatury do emulowania urządzenia terminala

Urządzenie terminala: dwie odsłony



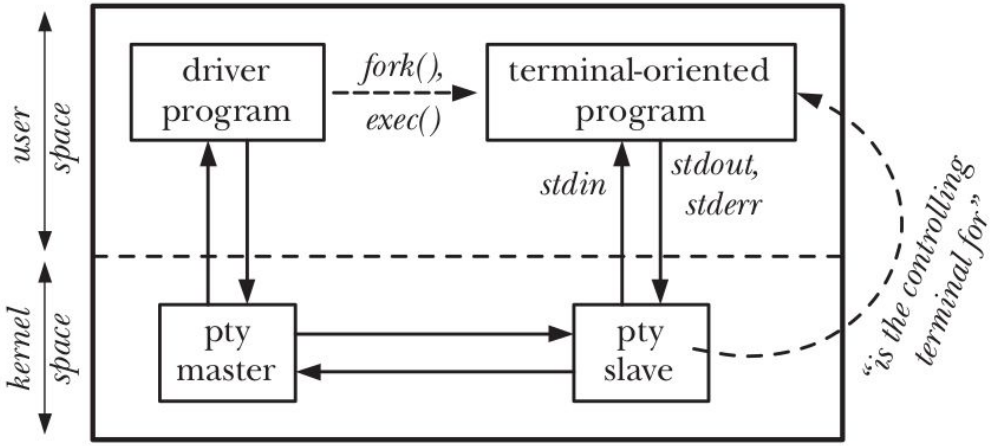
The TTY demystified – Linus Åkesson

Pseudoterminale: motywacja



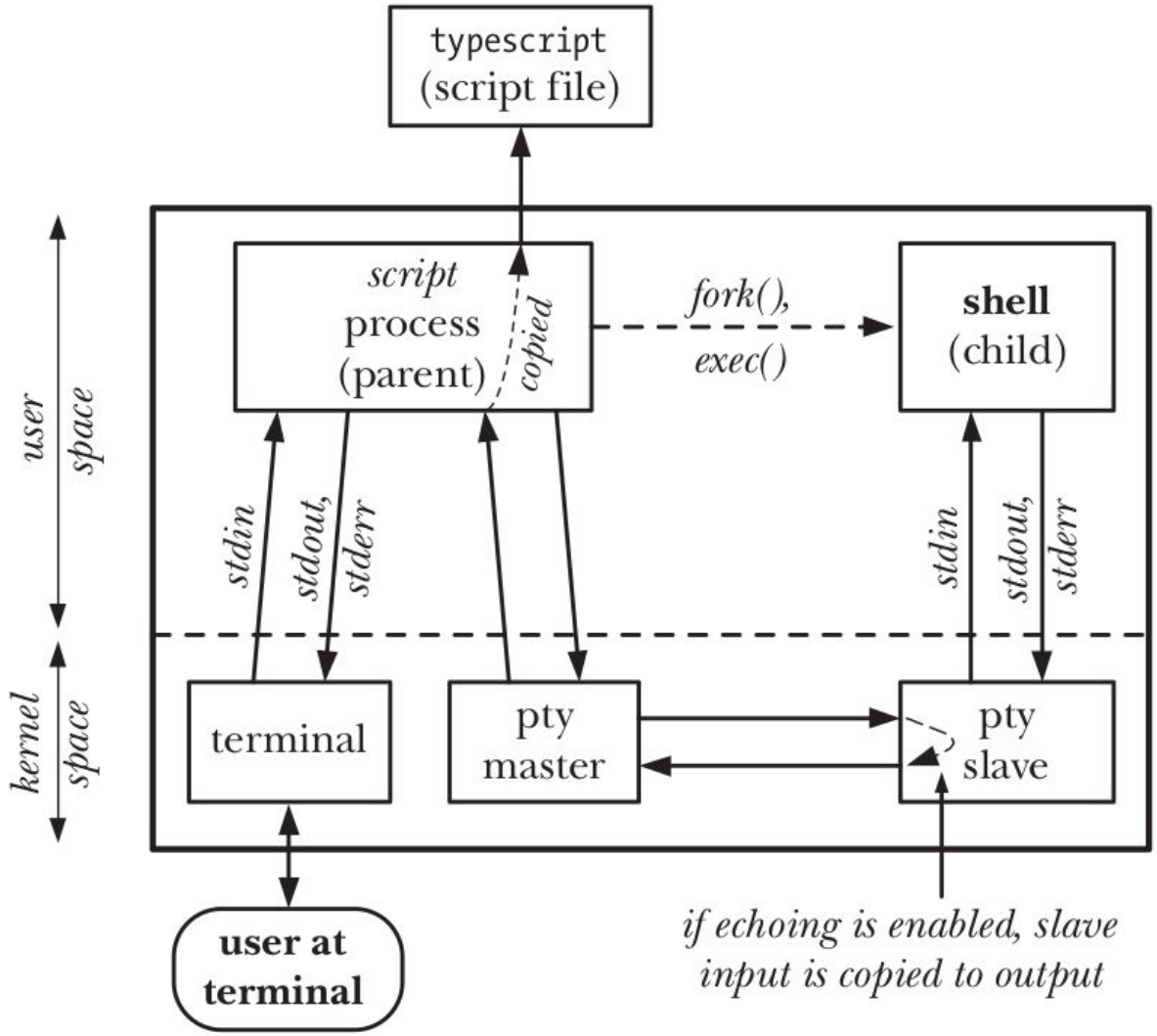
Chcielibyśmy, żeby to proces użytkownika (aka emulator terminala) odpowiadał za pobieranie znaków od użytkownika i wyświetlanie zawartości terminala – np. przy pomocy biblioteki do tworzenia GUI.

Pseudoterminal: właściwości

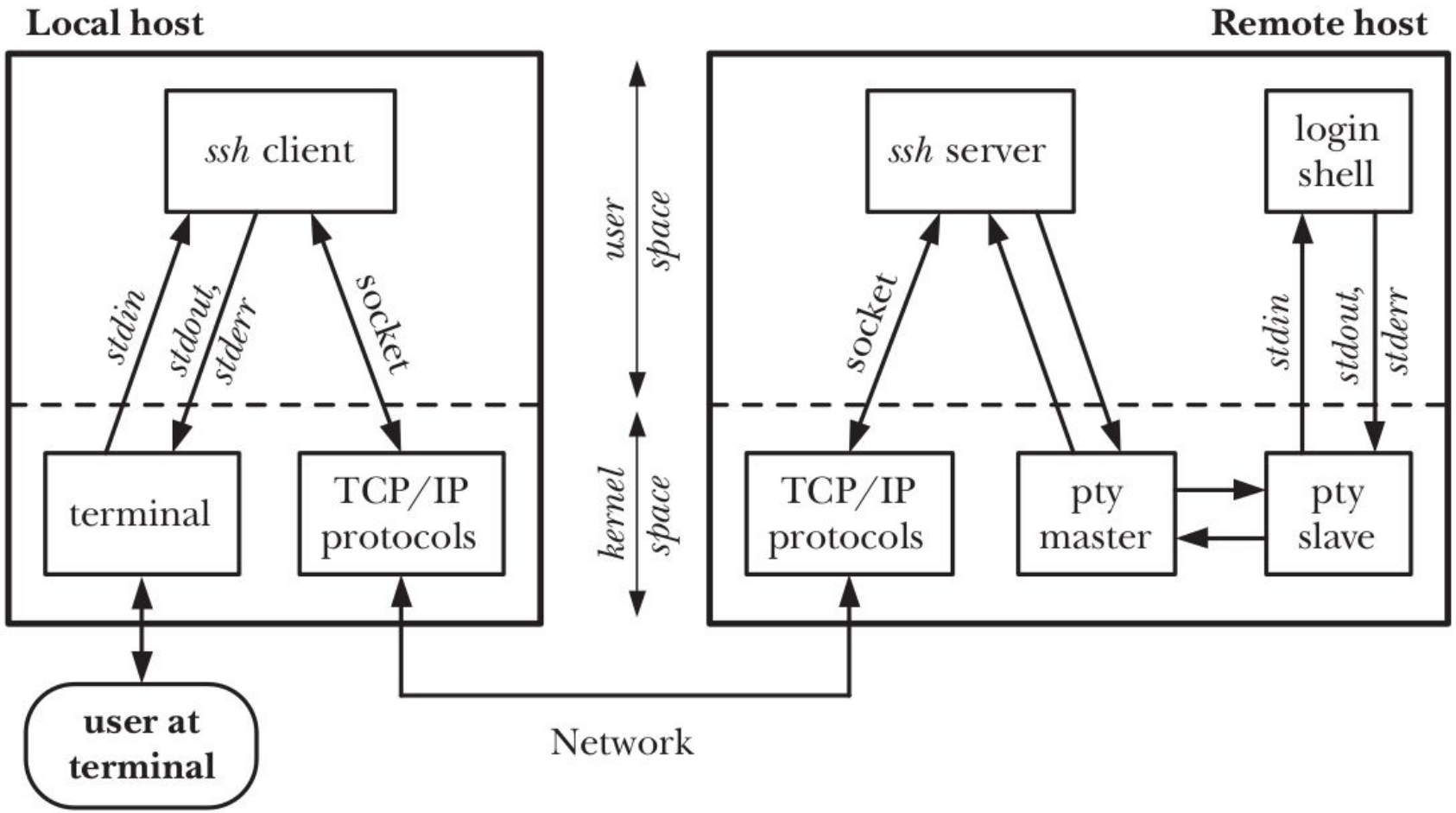


Funkcja openpty(3) tworzy dwa deskryptory plików odpowiadające końcom `master` i `slave`. Końcem `master` steruje program nadrzędny taki jak `xterm`. Koniec `slave` dostaje proces, który korzysta z terminala. Z punktu widzenia posiadacza `slave`, plik ten zachowuje się dokładnie tak samo jak prawdziwy plik urządzenia terminala.

`script(1)`: nagrywanie interakcji z terminalem



ssh(1): zdalna powłoka



Pytania?