Kurs administrowania systemem Linux

Zajęcia nr 7: Podstawowe czynności administracyjne w Linuksie

Instytut Informatyki Uniwersytetu Wrocławskiego

15 kwietnia 2024

Nazwy symboliczne i odpowiadające im numery

- Komputery posługują się wyłącznie liczbami (1, 2, 4, 8, 16-bajtowymi).
- Ludzie wolą nazwy symboliczne (napisy, często długie).
- Popularne przestrzenie nazw:
  - Nazwy hostów (np. `www.ii.uni.wroc.pl`).
  - Nazwy protokołów sieciowych (różnych warstw, np. `ip`, `icmp`, `udp`).
  - Nazwy serwisów (portów, np. `ssh`, `domain`, `http`).
  - Nazwy użytkowników (np. `root`).
  - Grupy użytkowników (np. `staff`, `adm`).
- Różne rodzaje serwisów określają relacje między nazwami symbolicznymi i numerami.
- W Linuksie dostępem do nich zarządza *Name Service Switch* (GNU C Library).

Name Service Switch (NSS)

Rodzaje serwisów

- files — Pliki tekstowe, zwykle w katalogu `/etc`.
- db — Bazy danych Berkeley DB, zwykle w `/var/db`. Szyszy dostęp, niż do plików testowych.
- nis — Network Information Service.
- nisplus — NIS+.
- dns — Domain Name Service (tylko dla nazw hostów).

Jest też kilka innych, zależnie od konfiguracji, np. compat lub ldap.

Name Service Switch (NSS)

serwis	zawartość	funkcja	plik w /etc
hosts	nazwy hostów i adresy IP	`gethostbyname(3)`	`hosts`
services	nazwy i numery portów sieciowych	`getservent(3)`	`services`
protocols	nazwy i numery protokołów sieciowych	`getprotoent(3)`	`protocols`
networks	nazwy sieci	`getnetent(3)`	`networks`
ethers	adresy MAC		`ethers`
aliases	aliasy pocztowe	`getaliasent(3)`	`aliases`
publickey	Secure RPC dla NFS i NIS+		`publickey`
rpc	nazwy i numery RPC	`getrpcbyname(3)`	`rpc`
passwd	informacje o użytkownikach	`getpwent(3)`	`passwd`
shadow	hasła użytkowników	`getspnam(3)`	`shadow`
group	grupy podstawowe użytkowników	`getgrent(3)`	`group`
initgroups	grupy dodatkowe użytkowników	`getgrouplist(3)`	`group`
netgroup	grupy użytkowników w sieci		`netgroup`

Plik nsswitch.conf(5)

nsswitch.conf

```
passwd:         compat nisplus
group:          compat nisplus
shadow:         compat nisplus
gshadow:        files nisplus
hosts:          files dns
networks:       files
protocols:      db files
services:       db files
rpc:            db files
netgroup:       nis
```

- Zob. też nss(5).
- Wiele programów ma opcję -n, która wyłącza usługę NSS.
- Odpytywanie: getent(1). Por. getent hosts localhost oraz np. dig localhost.

Użytkownicy i grupy

- Baza informacji o użytkownikach (lokalnych): `/etc/passwd`
- Baza haseł: `/etc/shadow` (tylko dla roota)
- Baza informacji o grupach użytkowników: `/etc/group`
- Logi: `/var/log/{w,b}tmp`, `/var/run/utmp`, zob. `utmp(5)`, `utmpdump(1)`.
- Informacje o mnie: `id(1)`, `whoami(1)`, `logname(1)`, `groups(1)`.
- Informacje o innych: `w(1)`, `who(1)`, `pinky(1)` (d. `finger(1)`), `users(1)`, `last(1)`.
- Zob. też `who am i`, `who mom likes` itp.
- Wiele grup zezwalających na dostęp do urządzeń: `cdrom`, `floppy`, `dialout`, `bluetooth`, `audio`, `video`, `wireshark`, `kvm`, `plugdev`, `netdev` i in.
- ... i wykonywanie czynności: `staff`, `operator`, `adm` itd.
- Zwykle instalator traktuje pierwszego konfigurowanego użytkownika jako szczególnie uprawnionego.
- System weryfikacji uprawnień jest dosyć szczelny. Warto tworzyć i używać konta w celu separacji dostępu do danych (por. `lp`, `mail`, `irc`, `nobody` itd.). Oczywiście piaskownice są bardziej szczelne.

Plik /etc/passwd (zob. passwd(5))

Każdy wpis zajmuje jeden wiersz, 7 pól oddzielonych znakiem „:”

1. nazwa użytkownika (*login name*)
2. zaszyfrowane hasło, znak `x` (por. `/etc/shadow`) lub puste
3. numer użytkownika (w Linuksie ≥ 1000 dla zwykłych użytkowników)
4. numer grupy głównej użytkownika (por. `/etc/group`)
5. pole GECOS (komentarz)
6. katalog domowy użytkownika
7. powłoka startowa użytkownika (opcjonalnie, por. `chsh(1)`)

Pole GECOS (General Electric Comprehensive Operating Supervisor 1962), 5 pól oddzielonych przecinkami (por. `chfn(1)` i `login.defs(5)`).

1. imię i nazwisko lub nazwa programu (`f`)
2. numer pokoju (`r`)
3. numer służbowego telefonu (`w`)
4. numer prywatnego telefonu (`h`)
5. dodatkowe informacje kontaktowe (`o`)

/etc/shadow, zob. shadow(5)

9 pól w formacie `/etc/passwd`. Czasy w sekundach epoki Uniksa.

1. nazwa użytkownika (*login name*)
2. zaszyfrowane hasło (ew. poprzedzone `!` lub `*`) lub puste
3. data ostatniej zmiany hasła
4. minimalny wiek hasła do zmiany
5. maksymalny wiek hasła do zmiany ($<$ poprz., zmiana zablokowana)
6. okres ostrzegania o konieczności zmiany hasła
7. okres możliwości zalogowania z wymuszeniem zmiany hasła po wygaśnięciu jego ważności
8. data wygaśnięcia konta (jeśli 0, tj. 1/1/1970, konto zablokowane)
9. pole zarezerwowane

Dodatkowo pliki:

- `/etc/{passwd-,shadow-,group-,gshadow-,subuid-,subgid-}` — zawartość plików sprzed ostatniej zmiany
- `/var/backups/{passwd,shadow,group,gshadow}.bak` — periodyczne kopie zapasowe (zob. `/etc/cron.daily/passwd`)

Grupy, podużytkownicy i podgrupy

Grupy

- Plik `/etc/group` — 3 pola: nazwa grupy, hasło, lista użytkowników.
- Hasła do grup zwykle w `/etc/gshadow`. Wówczas także możliwość zdefiniowania administratorów grup.
- Można być członkiem grupy lub mieć hasło do grupy.
- Polecenie `newgrp(1)`.
- Polecenia `su(1)` i `sg(1)`.

Podużytkownicy i podgrupy

- Pliki `/etc/{subuid,subgid}`
- Potrzebne np. przy uruchamianiu kontenerów nieuprzywilejowanych.

Jak zmienić zapomniane hasło roota?

Zwykle działa

- Uruchom system ratunkowy, np. z pendrive'a.
- Zamontuj *rootfs* systemu ratowanego np. w `/target/`.
- Pierwszy wiersz `/target/etc/passwd` zmień na `root::0:0:root:/root:/bin/bash`
- Uruchom system ratowany.
- Zaloguj się na konto `root` podając puste hasło.
- Ustaw nowe hasło roota poleceniem `passwd(1)`.

Warianty

- Usunąć hasło z `/etc/shadow`.
- W-chroot-ować się w system ratowany i wykonać polecenie `passwd(1)`.
- Komplikacje: hasła do BIOS-u, dysku itp.

Morał

- W razie fizycznego dostępu do komputera hasło roota nie jest zabezpieczeniem.
- Rootfs powinien się znajdować na zaszyfrowanej partycji.

Narzędzia do zarządzania użytkownikami

Klasyczne rozwiązania (Unix, Linux)

- Zamiast ręcznie edytować `/etc/passwd` itd. — specjalne programy.
- Bezpieczna edycja plików systemowych: `vipw(8)`, `vigr(8)` (także `visudo(8)`).
- Niskopoziomowe narzędzia `useradd(8)`, `userdel(8)`, `usermod(8)` — kompleksowe zmiany w plikach `/etc` i katalogu `/home`.
- Zakładanie wielu użytkowników na raz: `newusers(8)`.

W Debianie

- Pakiety: `passwd`, `shadow-utils` i `adduser`.
- Narzędzia Debiana: `adduser(8)`, `deluser(8)`, `addgroup(8)`, `delgroup(8)`.
- Konfiguracja w `adduser(5)`, `deluser(5)`.

Zarządzanie użytkownikami

- Dodanie użytkownika (Debian): `adduser` *`user`*
- Dodanie użytkownika do grupy (Debian): `adduser` *`user group`*
  Uwaga: użytkownik *`user`* będzie należał do tej grupy w sesji logowania rozpoczętej po wykonaniu tego polecenia — trzeba się wylogować i zalogować.
- Zablokowanie użytkownika *`user`*: `usermod -e 1970-01-01` *`user`*.
- Odblokowanie użytkownika *`user`*: `usermod -e` *`user`*
- Zablokowanie/odblokowanie *hasła* użytkownika *`user`*: `passwd [-l | -u] user`
- Zmiana hasła użytkownika *`user`*: `passwd` *`user`*
- Zmiana czasów ważności hasła: `chage(1)`
- Wykonanie powłoki jako użytkownik *`user`*: `su -` *`user`*
- Wykonanie programu w podanej grupie: `sg` *`grupa program`*

Ograniczenia dostępu do konta

Pełna blokada

- Użytkownik zablokowany — nie można uruchomić procesu z UID tego użytkownika.

Ograniczenia

- Domyślna powłoka zablokowana (np. `false(1)`, `nologin(1)`) — nie można się zalogować w sytuacjach, w których system wymusza użycie domyślnej powłoki (np. logowanie na konsoli, zdalnie poprzez ssh itp.). Zastosowania:
  - użytkownicy systemowi (np. demony),
  - dostęp do konta poprzez inne protokoły, np. `ftp`.
- Domyślna powłoka ograniczona (np. `rbash(1)`) — można się zalogować, ale zbiór dostępnych poleceń jest ograniczony.
- Hasło zablokowane — nie można się uwierzytelnić za pomocą hasła. Zastosowania:
  - użytkownicy systemowi (np. demony),
  - użytkownik uwierzytelnia się w inny sposób (zdalnie bądź lokalnie — certyfikaty, tokeny itp.).
- Dostęp zdalny zablokowany — zob. np. `sshd_config(5)`. Użytkownik loguje się lokalnie (terminal, `su` itp.). Zastosowania: np. konto `root`.

/etc/sudoers

- `sudo` — selektywne nadawanie uprawnień do wykonywania jako root pojedynczych programów.
- W Debianie pakiet `sudo`.
- Baza danych: plik `/etc/sudoers`, zob. `sudoers(5)`.
- Nie modyfikować zwykłym edytorem! Program `visudo(8)`: brak hazardów czasowych (zakłada locka) i pozostawiania kopii zapasowych. Sprawdza poprawność składniową pliku przy zapisie.
- Także `sudoedit`, `sudo -e` — edycja plików zamiast wykonywania programów.

Składnia w skrócie

kto skąd=(*jako-kto*:*z-jaką-grupą*) *co-wykonać*

- `ALL` oznacza wzorzec pasujący do wszystkiego.
- Przykład: `jan localhost=(root:staff) /bin/ip`
  — `jan` może uruchomić `ip(8)` jako `root` w grupie `staff`.
- Używać bezwzględnych ścieżek do programów!

Program sudo

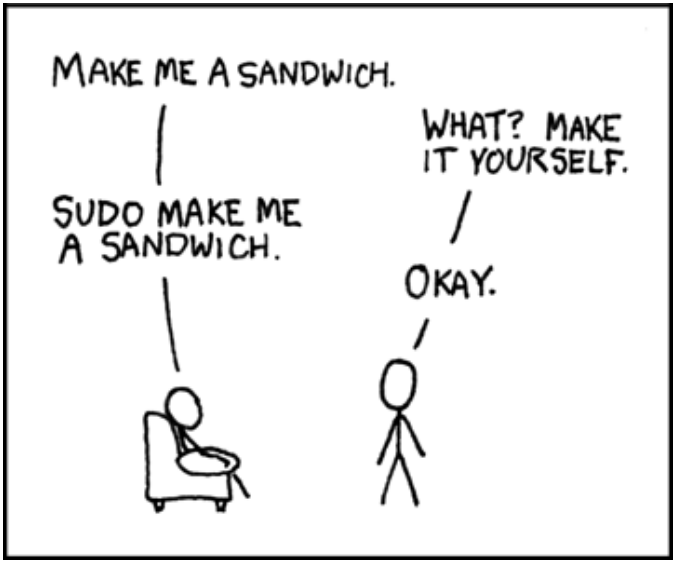
Wykonanie pojedynczego polecenia jako root

- `su - -c` *`polecenie`* — wymaga podania hasła *roota*.
- `sudo` *`polecenie`* — wymaga *jednorazowego* podania hasła *użytkownika*.

Własności sudo

- Użytkownik nie musi znać hasła roota.
- Hasło roota może w ogóle być wyłączone (por. Ubuntu).
- `sudo [-u` *`user`*`] -i` — uruchomienie powłoki jako użytkownik *`user`*.
  Lepsze niż `sudo su` lub `sudo /bin/bash`.
- Uwaga: `sudo` przydaje się w skyptach! (zob. także opcję `-n`)
- Pamiętaj o opcji `-k`, `-K`.
- W Ubuntu był *exploit* na `sudo -k`.
- Nie używaj bez potrzeby opcji `:NOPASSWD`!

XKCD sudo


MAKE ME A SANDWICH.
WHAT? MAKE IT YOURSELF.
SUDO MAKE ME A SANDWICH.
OKAY.

Hasło roota?

Czy blokować?

- Wszystko, co nie jest używane, powinno być zablokowane.
- W niektórych dystrybucjach domyślnie hasło roota jest wyłączone.
- Instalator Debiana pyta, choć sugeruje, żeby pozostawić włączone.
- Można zablokować: `passwd -l`, a jak się nie spodoba — odblokować: `sudo passwd -u`.
- Zawsze można zresetować, jeśli nawet się zapomni.
- Uwaga: jedyne hasła, których *absolutnie nie wolno* zapomnieć, to hasła do kryptografii (zaszyfrowane partycje itp.).

Krytyka sudo

- Program bardzo duży i skomplikowany.
- Skomplikowany plik konfiguracyjny — ryzyko błędnego skonfigurowania.
- Wykryto poważne podatności, zob. np. Animesh Jain: CVE-2021-3156: Heap-Based Buffer Overflow in Sudo (Baron Samedit).
- W OpenBSD `doas(1)`, zob. Ted Unangst: `https://flak.tedunangst.com/post/doas`.

Logi systemowe

- Kopalnia wiedzy o systemie.
- Warto je stale przeglądać i analizować.
- Katalog `/var/log/`.
- Większość plików do odczytu dla grupy `adm` — warto dodać siebie do tej grupy, by móc przeglądać logi jako zwykły użytkownik.
- Klasycznie: demon `(r)syslog`, zob. `rsyslog.conf(5)`, `rsyslogd(8)`.
- W systemd: `journalctl(1)`.
- Polecenie `logger(1)`.
- Automatyczne usuwanie starych logów: `logrotate(8)`.
- Warto wydłużyć „czas życia" logów w `/etc/logrotate.conf`, `/etc/logrotate.d/`.
- Programy `ccze(1)`, `clog(1)`, `colortail(1)`, `lwatch(1)` itp.

Wielozadaniowy system operacyjny

System operacyjny

- Zapewnia abstrakcję i wirtualizację *hardware'u*:
  - procesora i pamięci: system wielozadaniowy z podziałem czasu i pamięcią wirtualną;
  - pamięci masowej: system plików.
- Wirtualizacja wymaga wsparcia sprzętowego:
  - wirtualizacja procesora: tryb nadzorcy, wywołania systemowe;
  - wirtualizacja pamięci: ochrona i zarządzanie pamięcią (MMU, *Memory Management Unit*).

Procesy

- Programy działające w zwirtualizowanym środowisku.
- *Przełączanie kontekstów* daje wrażenie wyłącznego i nieprzerwanego dostępu do procesora.
- *Adresy wirtualne* dają wrażenie dostępu do ciągłej przestrzeni adresowej wielkiego rozmiaru (np. $2^{32}\,\mathrm{B} = 4\,\mathrm{GiB}$ w IA32, np. $2^{36}\,\mathrm{B} = 64\,\mathrm{GiB}$ w IA32 PAE, $2^{48}\,\mathrm{B} = 256\,\mathrm{PiB}$ w x86-64, $2^{57}\,\mathrm{B} = 4\,\mathrm{EiB}$ w x86-64 5-level — Sunny Cove).
- *Wywołania systemowe* (*syscall*) dają abstrakcję przerwań.

Procesy i wątki

- Każdy proces ma własny wirtualny procesor i własną wirtualną pamięć.
- Proces może się składać z jednego lub wielu *wątków*.
- Wątki mają własne wirtualne procesory (w tym własne stosy wywołań, liczniki rozkazów itp.), ale w ramach jednego procesu współdzielą jego pamięć wirtualną (w tym zmienne statyczne, stertę itd.).
- Kod jądra jest wykonywany w trybie nadzorcy, zwykłe procesy — w trybie użytkownika.
- Jądro wykonuje wiele wątków jednocześnie.
- Wszystkie wątki jądra współdzielą pamięć (jądro monolityczne).
- Procesy mogą uruchamiać nowe procesy (`fork(2)`) i wątki (`clone(2)`).
- Każdy proces ma dokładnie jednego ojca (jest jedno *drzewo procesów*).
- Przodkiem każdego procesu w przestrzeni użytkownika jest `init(1)`. Ojcem każdego wątka jądra jest `kthreadd`.

Numery procesów

- Każdy wykonywany kod podlegający podziałowi czasu ma unikatowy numer PID (*Process ID*):
  - każdy wątek jądra,
  - każdy proces (współdzieli numer z pierwszym wątkiem),
  - każdy dodatkowy wątek procesu.
- $0 \leq \text{PID} \leq$ `PID_MAX`.
- PID jest tradycyjnie liczbą całkowitą ze znakiem. Dawniej dwubajtową, stąd dawniej przeważnie `PID_MAX` $= 32767$.
- Dawniej `PID_MAX` — konfiguracja kompilacji jądra, w jądrze 2.6 i później sysctl `kernel.pid_max` równy `PID_MAX` $+ 1$. W IA-32 co najwyżej 32768, w x86-64: 4 Mi (2^{22}).
- Numery przydzielane są po kolei, a po osiągnięciu `PID_MAX` wyszukiwanie wolnych numerów zaczyna się od 300 (karencja).
- Umownie PID 0 — `idle` (wykonywany, gdy żaden proces bądź wątek nie jest wykonywany). Formalnie jest ojcem `init(1)` (PID 1) oraz `kthreadd` (PID 2).
- PID ojca procesu nazywa się PPID (*Parent PID*).
- W Bashu zmienne środowiskowe `$`, `BASHPID` i `PPID`.

Cykl życia procesów

- `init(1)` (PID 1) żyje przez cały czas pracy systemu.
- Każdy inny proces jest uruchamiany za pomocą `fork(2)`.
- Ojcem procesu jest ten, kto wywołał `fork(2)`.
- PCB (*Process Control Block*) — struktura w jądrze opisująca proces.
- PCB pozostaje po zakończonym procesie i zawiera m. in. *kod powrotu*.
- Ojciec ma obowiązek „pochować” zmarłego syna (`wait(2)`, `waitpid(2)`, `waitid(2)`) lub jawnie zignorować jego śmierć (np. `SIG_IGN` lub `SA_NOCLDWAIT` dla `SIGCHLD`, zob. `wait(2)`, `sigaction(2)`), w przeciwnym razie zmarły syn staje się *zombie* i tkwi w tablicy procesów.
- *Reparenting*: jeśli ojciec umrze wcześniej niż syn, to syn jest automatycznie adoptowany przez `init(1)`.
- `init(1)` periodycznie dokonuje pochówku wszystkich zmarłych synów.
- Niektóre procesy przed śmiercią (nie dotyczy nagłej śmierci poprzez `SIGKILL`) wysyłają np. sygnał `SIGTERM` nie dopuszczając, by dzieci przeżyły ojca.

Uzyskiwanie informacji o procesach

`ps(1)`

- Wiele opcji w trzech wersjach: BSD, standard i GNU.
- Wypisanie wszystkich procesów (`-e`) w kolumnach `UID PID PPID C STIME TTY TIME CMD` (`-f`) bez skracania wierszy (`-w -w`):
  `ps -efww`
- Wypisanie wszystkich procesów z pominięciem wątków jądra:
  `ps -fN --ppid 2 --pid 2`

`pstree(1)`

- Wypisuje ładnie sformatowane drzewo procesów.
- Wypróbujcie `pstree -lpgUSuna | less`

`top(1)`

- Dynamicznie (domyślnie co 3 sekundy) wyświetla tabelę procesów i statystyki.
- Pełnoekranowy z kolorami. Zob. też `htop`.

Użytkownicy i grupy

Każdy proces ma przypisanych czterech użytkowników i grupy:

- real — kto uruchomił proces,
- effective — czyje prawa dostępu ma proces,
- saved — effective UID/GID z chwili uruchomienia procesu,
- filesystem — czyje prawa dostępu do plików ma proces.

Można zmieniać UID/GID:

- `{s,g}et{,e,re,res,fs}{u,g}id(2)` zmieniają/ujawniają real/effective/real+effective/real+effective+saved/filesystem UID/GID procesu.
- `su(1)`, `sg(1)` uruchamiają proces z podanym real UID/GID.
- `newgrp(1)` zmienia real GID powłoki.
- `id(1)` ujawnia real i effective UID użytkownika, `whoami(1)` — tylko effective.

Sygnały

- Mechanizm asynchronicznego przesyłania komunikatów pomiędzy procesami lub procesami i jądrem.
- Sygnały mają numery (liczby typu `int`) i przyporządkowane im nazwy (`signal(7)`, `bits/signum.h`).
- Każdy proces może rejestrować procedury wywoływane w razie otrzymania sygnału (`sigaction(2)`).
- Każdy proces może wysłać sygnał do innego procesu (`kill(2)`, obwoluta: `kill(1)`).
- Jądro wysyła sygnały do procesu (np. `SIGHUP`, `SIGINT`, `SIGQUIT`, `SIGILL`, `SIGFPE`, `SIGSEGV`, `SIGPIPE` itd.).
- Jądro przechwytuje sygnały kierowane do procesu: `SIGKILL`, `SIGSTOP`, `SIGCONT`.
- Jądro przesyła do `init(1)` tylko te sygnały, dla których `init` zarejestrował *handlery* (zabezpieczenie przed przypadkowym zabiciem `init`).

Sygnały we współczesnym Linuksie (x86)

1 SIGHUP	12 SIGUSR2	23 SIGURG
2 SIGINT ^C	13 SIGPIPE	24 SIGXCPU
3 SIGQUIT ^\	14 SIGALRM	25 SIGXFSZ
4 SIGILL	15 SIGTERM	26 SIGVTALRM
5 SIGTRAP	16 SIGSTKFLT	27 SIGPROF
6 SIGABRT	17 SIGCHLD	28 SIGWINCH
7 SIGBUS	18 SIGCONT	29 SIGIO
8 SIGFPE	19 SIGSTOP	30 SIGPWR
9 SIGKILL	20 SIGTSTP ^Z	31 SIGSYS
10 SIGUSR1	21 SIGTTIN	32-63 real-time signals
11 SIGSEGV	22 SIGTTOU	

Szczegóły: zob. signal(7).

Wysyłanie sygnałów z powłoki

- Program `kill(1)` oraz *shell builtin* `kill` różniące się opcjami.
- `kill -l` wypisuje dostępne sygnały.
- `kill [-`*`SIG`*`] `*`PID`* wysyła sygnał *SIG* (domyślnie `TERM`) do procesu *PID* (*PID* $= 0$ oznacza wszystkie procesy z grupy procesu wysyłającego (włączając ten proces), *PID* $= -1$ oznacza wszystkie procesy z wyjątkiem siebie i `init`, *PID* < -1 oznacza wysłanie do grupy $-$*PID*).
- Programy `killall(1)`, `killall(5)` — *kill by name*.
- Programy `pgrep(1)` i `pkill(1)` — fuzja `grep(1)` i `killall(1)`.
- Program `pidof(1)`.

Sesje

- Zbiory procesów przypisane do jednego terminala.
- Sesje mogą też dziedziczyć terminal bądź nie być przypisane do terminala.
- *Syscall* `setsid(2)` i program `setsid(1)` tworzą nową sesję.
- Podczas *logowania* (zob. `login(1)`) jest tworzona nowa *sesja*, związana z terminalem sterującym (*controlling terminal*).
- Terminalem może być urządzenie transmisji szeregowej RS-232 (`ttyS{0..}`) albo USB (`ttyUSB{0..}`), terminal wirtualny (`tty{0..}` tworzony przez jądro w trybie tekstowym karty graficznej lub poprzez KMS) bądź pseudoterminal (`pts/{0..}`) tworzony np. przez aplikację terminala w systemie okienkowym bądź poprzez zdalne logowanie (np. `sshd(8)`).
- Domyślnie wszystkie procesy w sesji mają `stdin`, `stdout` i `stderr` związane z terminalem sterującym.

Sesje, cd.

- Sesja ma lidera, którym jest zwykle powłoka systemowa podłączona do tego terminala.
- Jeśli terminal sterujący zostanie rozłączony (*hangup*) lub lider sesji umrze, wszystkie procesy w sesji otrzymują sygnał `SIGHUP`.
- Program `nohup(1)` pozwala uruchomić proces w osobnej sesji nie połączonej z terminalem.
- GNU `screen(1)` i BSD `tmux(1)` pozwalają na uruchomienie sesji, które mogą być wielokrotnie podłączane i odłączane od terminali.