Wrocław
University
of Science
and Technology

Administrowanie sieciowymi systemami operacyjnymi

Wykład 2

Sieciowy system operacyjny, sterowniki, moduły, powłoka

dr inż. Jarosław Rudy

15 marca 2023

HR EXCELLENCE IN RESEARCH

Sieciowy i rozproszony system operacyjny

Sieciowy system operacyjny (1)

System operacyjny z zintegrowanym w jądrze stosem sieciowym.
Równoważnie, system operacyjny umożliwiający pracę w sieci komputerowej.

Sieciowy system operacyjny (2)

Wyspecjalizowany system operacyjny do stosowania jako oprogramowanie w urządzeniach sieciowych (przełączniki, rutery itp.).

Rozproszony system operacyjny

Zespół osobnych komputerów (węzłów) wraz z ich oprogramowaniem systemowym, udostępniający swoje zasoby i widoczny dla użytkownika jak jednolity (zwykły) system operacyjny.

Sieciowy i rozproszony system operacyjny

Sieciowy system operacyjny (1)

System operacyjny z zintegrowanym w jądrze stosem sieciowym.
Równoważnie, system operacyjny umożliwiający pracę w sieci komputerowej.

System zawiera:

- sterowniki obsługujące interfejsy (karty) sieciowe,
- wbudowane oprogramowanie obsługujące co najmniej warstwy łącza danych, sieciowej i transportowej,
- funkcje systemowe/biblioteczne wspierające API sieciowe w postaci np.:
  - gniazd sieciowych Berkeley (znane też jako gniazda POSIX lub BSD),
  - gniazd sieciowych Windowsa (WSA, Winsock),
  - Transport Layer Interface (dla Solaris, „Classic" MacOS).

Sieciowy i rozproszony system operacyjny

Sieciowy system operacyjny (1)

System operacyjny z zintegrowanym w jądrze stosem sieciowym.
Równoważnie, system operacyjny umożliwiający pracę w sieci komputerowej.

Od lat osiemdziesiątych dwudziestego wieku systemy operacyjne powszechnie wspierały interfejsy sieciowe do tego stopnia, że obecnie termin „system operacyjny" prawie zawsze oznacza „sieciowy system operacyjny". Przykłady:

- Systemy uniksopochodne (BSD, System V i ich pochodne, Solaris, macOS, iOS itd.),
- Systemy uniksopodobne, FreeBSD, różne dystrybucje GNU/Linux, Android,
- Windows 9x, począwszy od Windows 95,
- Windows NT (Windows XP, 7, Vista, 10),

Sieciowy system operacyjny w powyższym znaczeniu będzie głównym tematem kursu (na przykładzie Debian GNU/Linux).

Sieciowy i rozproszony system operacyjny

Sieciowy system operacyjny (2)

Wyspecjalizowany system operacyjny do stosowania jako oprogramowanie w urządzeniach sieciowych (przełączniki, rutery itp.).

Przykładowe sieciowe systemy operacyjne w powyższym sensie:

- Cisco Internetwork Operating System (Cisco Systems)
- ZyNOS (Zyxel Communications),
- RouterOS (MikroTik), bazowany na Linuksie,
- DD-WRT, bazowany na Linuksie,
- Dell Networking Operating System, DNOS (Dell Networking), bazowany na NetBSD/Linux, zależnie od wersji.

Sieciowy i rozproszony system operacyjny

Rozproszony system operacyjny

Zespół osobnych komputerów (węzłów) wraz z ich oprogramowaniem systemowym, udostępniający swoje zasoby i widoczny dla użytkownika jak jednolity (zwykły) system operacyjny.

Przykładowe rozproszone systemy operacyjne:

- Inferno (Bell Labs),
- Plan 9 from Bell Labs,
- Amoeba,
- QNX (BlackBerry Limited). Uniksopodobny system wbudowany, czasu rzeczywistego o twardych ograniczeniach (hard real-time system).

Sieciowy system operacyjny i administracja

- system operacyjny (bez elementów sieciowych).
  - ~~procesy, przydział czasu procesora, pamięć wirtualna, IPC,~~
  - ~~sterowniki, sprzęt,~~
  - ~~współbieżność, wątki,~~
  - system plików,
  - użytkownicy, uprawnienia, autoryzacja,
  - powłoka,
  - konfiguracja, administracja,

Sieciowy system operacyjny i administracja

- system operacyjny (bez elementów sieciowych).
  - ~~procesy, przydział czasu procesora, pamięć wirtualna, IPC,~~
  - ~~sterowniki, sprzęt,~~
  - ~~współbieżność, wątki,~~
  - system plików,
  - użytkownicy, uprawnienia, autoryzacja,
  - powłoka,
  - konfiguracja, administracja,
- sieciowy system operacyjny,
  - stos sieciowy,
  - usługi sieciowe,
  - konfiguracja, administracja.

Sterowniki

- Część jądra systemu operacyjnego odpowiedzialna za obsługę urządzenia wejścia-wyjścia (ogólnie urządzenia peryferyjnego).
- Dostarcza abstrakcyjny interfejs do obsługi urządzenia, bez konieczności znajomości jego szczegółów. Czasami jeden sterownik obsługuje wiele urządzeń.
- Podział:
  - urządzenia znakowe (character devices),
  - urządzenia blokowe (block devices),
  - interfejsy sieciowe (network interfaces).
- Sterowniki dołączane są do jądra statycznie (w czasie kompilacji) lub dynamicznie (poprzez moduły).

Sterowniki znakowe i blokowe

- Widoczne jako pliki specjalne w `/dev/` z własną ścieżką (np. `/dev/tty0`, `/dev/null`).
- Typ pliku (widoczny np. po użyciu `ls -l`) oznaczany jako `c` (urządzenia znakowe) lub `b` (urządzenia blokowe).
- Identyfikowane z wykorzystaniem liczby głównej i pobocznej:
  - liczba główna (major) identyfikuje sterownik,
  - liczba poboczna (minor) jest przekazywana do sterownika, identyfikuje konkretne urządzenie lub tryb pracy,
  - widoczne poprzez `ls -l` (zamiast rozmiaru pliku).
- Czasami jedno urządzenie dostępne jest zarówno przez urządzenie znakowe jak i blokowe.

Sterowniki znakowe i blokowe

- Lista numerów głównym znajduje się w `/proc/devices`. Przykładowy wynik:

```
Character devices:
  1 mem
  2 pty
  3 ttyp
  4 ttyS
  6 lp
  7 vcs
 10 misc
 13 input
 14 sound
 21 sg
180 usb

Block devices:
  2 fd
  8 sd
 11 sr
 65 sd
 66 sd
```

- Urządzenia znakowe i blokowe mają osobny spis liczb głównych tzn. urządzenie znakowe może mieć liczbę główną taką samą jak jakieś urządzenie blokowe.

Sterowniki znakowe i blokowe

- Pliki specjalne urządzeń (węzły, nodes) można tworzyć z użyciem komendy `mknod`. Składnia:

  ```
  mknod path type major minor
  ```

  gdzie:

  - `path` – ścieżka do pliku (np. `/dev/sda1`),
  - `type` – `c` lub `u` dla urządzeń znakowych, `b` dla urządzeń blokowych (`p` tworzy potok nazwany),
  - `major` – liczba główna,
  - `minor` – liczba poboczna.

- Użycie `mknod` wymaga praw roota.

Urządzenia znakowe

- Dane dostępne są sekwencyjnie (strumieniowo), znak (najczęściej bajt) za znakiem.
- Wskaźnik plikowy ma tylko jedną pozycję: aktualną. Ręczna zmiana położenia wskaźnika plikowego (seeking) jest zabroniona.
- Buforowanie nie jest wymagane (ale jest możliwe).
- Sterowniki urządzeń znakowych są prostsze, łatwiejsze w przygotowaniu i wymagają mniej uwagi.
- Zapis i odczyt są blokujące (wywołania `read` i `write` wracają gdy operacja się zakończy).
- Przykłady: porty szeregowe, porty równoległe, karty dźwiękowe, klawiatura, terminale.

Urządzenia blokowe

- Zawsze mają dostęp swobodny (random-access),
- Zwykle dotyczą fizycznych urządzeń, które zapisują i odczytują bloki o określonej długości.
- Dane są buforowane (obecność cache'a).
- Wsparcie dla zmiany pozycji wskaźnika plikowego (seeking).
- Sterowniki blokowe są trudniejsze w tworzenie i obsłudze.
- Żądania zapisu i odczytu wywoływane są przez system buforowania. Mogą być asynchroniczne.
- Przykłady: dyski twarde, pendrive'y, CD-ROM, DVD, kamery USB.

Interfejs urządzeń znakowych i blokowych

- Struktura `file_operations`:
  - `open`/`release` – otwarcie i zamknięcie urządzenia.
  - `read`/`write`.
  - `ioctl` – specyficzne funkcje urządzeń poza zapisem/odczytem.
  - `fsync` – synchronizacja bufora z urządzeniem (urządzenia blokowe).
  - `check_media_change`/`revalidate` – funkcje dla urządzeń z wymiennym nośnikiem. Sprawdzają czy nośnik się zmienił i uaktualniają odpowiednie informacje.
- `block_read`/`block_write` – odczyt/zapis wielu bloków (dostępne w osobnej strukturze).
- Kolejka żądań (m.in. funkcja `request_fn`) – dla urządzeń blokowych.

Interfejsy sieciowe

- Nie ma do nich dostępu przez ścieżkę (łamią zasadę „wszystko jest plikiem").
- Zamiast tego sterownik rejestruje swoją nazwę (np. eth0, enp0s31f6, wlp2s0, lo). Listę aktualnie dostępnych interfejsów można sprawdzić przy pomocy `ifconfig -a`.
- Sterowniki ukrywają zależność od konkretnego protokołu.
- Wiele otwartych gniazdek sieciowych może być podłączonych pod ten sam interfejs sieciowy.
- Operują na pojedynczych pakietach (podobne do urządzeń blokowych?).
- W odróżnieniu od innych urządzeń, interfejsy sieciowe reagują nie tylko na żądania z jądra (sterownika), ale również ze świata zewnętrznego (tzn. urządzenie żąda przesłania danych do jądra).

Rejestracja urządzeń

- Jądro identyfikuje sterowniki urządzeń blokowych i znakowych według zarejestrowanych numerów głównych:
  - Funkcje `register_chrdev` oraz `register_blkdev`.
  - Zakres liczb głównych: dawniej 0–127, obecnie 0-255 (zakres liczb pobocznych 0–255, obecnie presja na zwiększenie).
  - W starszych jądrach liczba główna jest narzucana statycznie przez ładowany moduł (z wykorzystaniem skryptu `makedev`).
  - Obecnie na ogół wybór dynamiczny: po podaniu liczby głównej 0 funkcja wybierze niezajętą liczbę i ją zwróci.
- Interfejsy sieciowe są rejestrowane funkcją `register_netdev`, konieczność podania dużej struktury `net_device` (zawiera m.in. nazwę interfejsu).

Moduły

- Fragmenty jądra (zwykle sterowniki) dołączane opcjonalnie.
- Mogą być dynamicznie ładowane i usuwane z jądra, bez konieczności restartu lub (długiej!) rekompilacji,
- Korzystają z wewnętrznego API jądra (jego podsystemów), a nie z wywołań systemowych (System Call Interface, SCI) przestrzeni użytkownika. Np. moduły korzystają z `printk` zamiast `printf`.
- Moduły pozwalają ograniczyć wielkość jądra (przydatne w niektórych zastosowaniach).
- Modularne jądro staje się bardziej elastyczne, wciąż pozostając monolityczne.
- Monolityczność sprawia jednak, że moduły muszą być starannie projektowane.
- Prawa autorskie i licencje w przypadku modułów, zwłaszcza własnościowych.

Moduły

- Pliki modułów (rozszerzenie `.ko`) zwykle umieszcza się w `/lib/modules`.
- Każdy moduł powinien implementować co najmniej funkcje `init_module` oraz `cleanup_module`. Funkcje te wywoływane są odpowiednio przy ładowaniu i usuwaniu modułu.
- Proste ładowanie modułu odbywa się poprzez komendę `insmod`.
- Ładowanie/usuwanie modułu z uwzględnieniem zależności odbywa się poprzez komendę `modprobe`.
- Listy zależności modułów są tworzone przez komendę `depmod`.
- Proste usuwanie modułu odbywa się poprzez użycie komendy `rmmod`.

Moduły

- Do modułów można przekazywać parametry:

```
insmod plik_modulu zm="my_dev" nr=81
```

- Listing modułów, wraz z ich użyciem dostępna jest przy użyciu komendy `lsmod`.
- Inny listing można uzyskać odczytując plik `/proc/modules`.
- Użycie demona `kerneld`:
  - Automatyzacja zarządzaniem i usuwaniem modułów.
  - Jądro musi być skompilowanie z wsparciem dla `kerneld`.
  - Otrzymuje od jądra żądania konkretnych modułów, identyfikatory ogólne.
  - Konfiguracja modułów w katalogu `/etc` (np. /etc/modules.conf), w tym aliasy.

Hierarchia procesów

- Procesy systemu operacyjnego startują z poziomu kodu jądra.
- Pierwszym procesem (`PID=1`) jest `init`:
  - proces uruchamiany jest ze ścieżki `/sbin/init`,
  - w praktyce `/sbin/init` często jako dowiązaniem symbolicznym (np. na popularny demon `systemd`),
  - wszystkie inne procesy są potomkami (bezpośrednimi lub pośrednimi) `init`. Staje się on też rodzicem osieroconych procesów.
- Każdy terminal ma proces jego obsługi (`getty`):
  - Proces `getty` wykonuje `exec /bin/login` wraz z naszym loginem,
  - Proces `login` przeprowadza uwierzytelnianie, ustawia środowisko (`UID`, katalog domowy, powłoka itp.) i wykonuje `exec powłoka`.
  - Procesy zalogowanych użytkowników stają się potomkami powłoki.

Konfiguracja init

- Konfiguracja znajduje się w `/etc` (np. `/etc/inittab`).
- Format wpisów w `inittab`:

  ```
  id:poziomy:akcja:proces
  ```

- Niektóre dostępne „akcje”:
  - `respawn`,
  - `wait`,
  - `once`,
  - `bootwait`,
  - `boot`,
  - `sysinit`,
  - `initdefault`.

Poziomy uruchomienia

- Typowe znaczenie poziomów 0–6 dla Linuksa:
  - 0 – halt,
  - 1 – jeden użytkownik (cele administracyjne),
  - 2 – wielu użytkowników, bez sieci,
  - 3 – standard bez GUI (wielu użytkowników, sieć),
  - 4 – zdefiniowany przez użytkownika,
  - 5 – podobnie jak poziom 3, ale dodaje GUI,
  - 6 – reboot.
- Poziom domyślny określony przez `initdefault`.
- Spotyka się też poziom S lub s (zwykle oznacza poziom 1 lub podobny).
- Do zmiany poziomu uruchomienia służy komenda `telinit`.

Przykład /etc/inittab

Przykładowa zawartość pliku `/etc/inittab` (źródło: [1]):

```
# inittab do Linuksa
id:1:initdefault:
rc::bootwait:/etc/rc
1:1:respawn:/etc/getty 9600 tty1
2:1:respawn:/etc/getty 9600 tty2
3:1:respawn:/etc/getty 9600 tty3
4:1:respawn:/etc/getty 9600 tty4
```

Skrypty startowe

- Skrypty startowe najczęściej umieszczone są w katalogu `/etc/init.d`.
- Skrypty te uruchamiane są z argumentem `start`, `stop`, `restart` lub `reload`.
- Które z powyższych skryptów należy wykonać na danym poziomie uruchomienia i w jakiej kolejności, precyzują katalogi `/etc/rcX.d` gdzie jest poziomem uruchomienia.
- Katalog `/etc/rcX.d` zawiera dowiązania do części skryptów z `/etc/init.d`, lecz ze zmienionymi nazwami:
  - pierwszy znak to `S` (start) lub `K` (kill/stop),
  - dwa kolejne znaki to numer (kolejność uruchomienia),
  - przykładowo `S19cron` oznacza uruchomienie skryptu `cron`.
- Komenda `update-rc.d` uaktualnia dowiązania `/etc/rcX.d`.

Powłoka i linia komend

Powłoka (1)

W szerokim znaczeniu jest to dowolne oprogramowanie pośredniczące pomiędzy użytkownikiem a systemem operacyjnym. Forma interfejsu dowolna: wierszowy (CLI), tekstowy (TUI), graficzny (GUI).

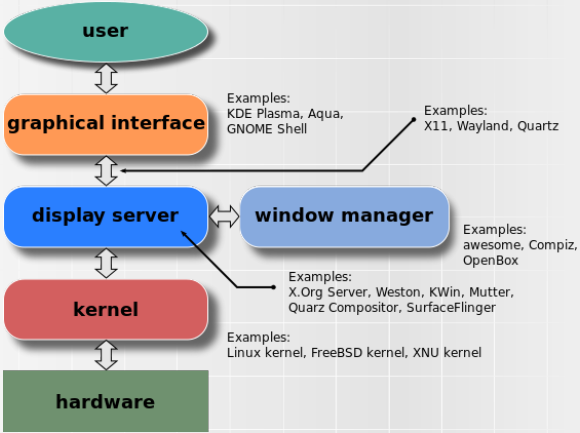
Powłoka (2)

Oprogramowanie pośredniczące pomiędzy użytkownikiem a systemem operacyjnym w postaci interfejsu wiersza poleceń.

Powłoka (3)

W wąskim znaczeniu to interpreter komend (skryptowy język programowania).

Powłoki graficzne



Źródło: [2]

Powłoki wiersza poleceń

Interfejs wiersza poleceń (Command-line Interface, CLI):

- terminale (np. `/dev/tty2`),
- pseudoterminale (np. `/dev/pts/0`),
- okna terminala (np. GNOME terminal, xterm).

Interpretery wiersza poleceń:

- Bourne shell (`sh`),
- Bourne-Again shell (`bash`),
- Korn shell (`ksh`),
- Almquist shell (`ash`) – podstawa dla powłok dla FreeBSD i NetBSD,
- Z shell (`zsh`) – dla macOS,
- C shell (`csh`) – bazowany na języku `C`.

Praca z powłoką

Pliki w katalogu użytkownika:

- `.profile` – skrypt uruchamiany przy logowaniu do systemu (login shells),
- `.bashrc` – skrypt uruchamiany przy „zwykłym" uruchomieniu shella,
- `.bash_logout` – skrypt uruchamiany przy opuszczeniu shella (np. wyczyszczenie konsoli),
- `.bash_history` – historia wykorzystanych komend.

Zmiana obecnej wirtualnej konsoli:

- `Alt + F1` do `Alt + F8` – zmiana konsoli głównej na `tt1` do `tt8`,
- Jeśli obecna konsola jest graficzna (zwykle `tt7`), to trzeba do kombinacji dodać `Ctrl`.
- `chvt` – zmiana za pomocą komendy.
- `tty` – wyświetl urządzenie aktualnego terminala.

Praca z powłoką

Skróty klawiszowe:

- `Ctrl+D` – zasymulowanie końca pliku. Przydatne do kończenia pracy z niektórymi komendami (w tym `bash`),
- `Ctrl+C` – wysłanie sygnału przerwania (`SIGINT`). Zabija proces, jeśli sygnał nie jest inaczej obsłużony,
- `Ctrl+Z` – wysłanie sygnał zatrzymania procesu (`SIGTSTP`). Warto zapoznać się też z komendami `fg`, `bg`, `jobs` oraz mechanizmem `&`),
- `Ctrl+W` – skasowanie poprzedniego słowa,
- `Ctrl+Shift+C` oraz `Ctrl+Shift+V` – kopiuj/wklej (terminale graficzne),
- `Ctrl+U` oraz `Ctrl+Y` – wytnij oraz wklej obecny wiersz.
- Kursor w górę i w dół – ostatnie komendy.

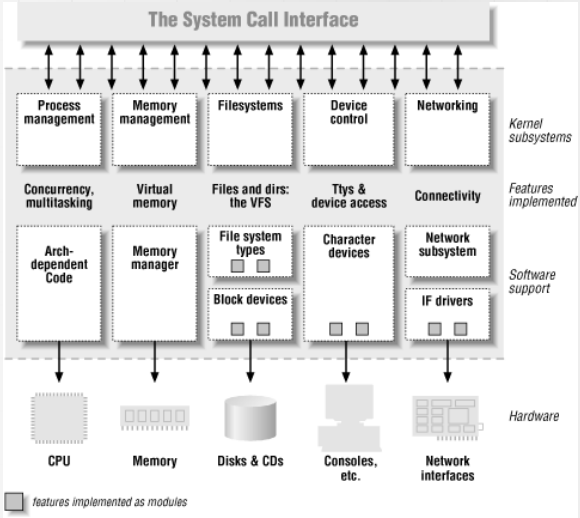
Praca z powłoką

- `cd` – zmiana aktualnego katalogu (Current Working Directory, CWD).
  - bez argumentu `cd` przechodzi do katalogu domowego,
  - możemy przejść do poprzedniego katalogu, podając jako argument `-`.
- Przekierowania:
  - przekierowanie na wejście pliku (`<`) lub tekstu (`<<<`),
  - przekierowaniu wejścia do pliku (`>` oraz `>>`),
  - przekierowanie konkretnych deskryptorów (`2 >` oraz `2 >&1`),
  - połączenie komend potokiem (`|`),
  - wykonanie warunkowe (`&&` oraz `||`),
  - wykonanie w tle (`&`).

Praca z powłoką

- podgląd procesów (`top`, `ps`),
- wyświetlanie plików (`cat`, `head`, `tail`, `less`, `more`, `most`),
- edytory tekstu (`nano`, `emacs`, `vi(m)`),
- `screen` – multiplekser terminala,
- `script` – zapis przebiegu sesji (wraz z kolorami),
- dokumentacje (`man`, manuale online np. `die.net`).

Podsumowanie



Źródło: [3]

Bibliografia


https://manpages.debian.org/testing/sysvinit-core/inittab.5.en.html


https://commons.wikimedia.org/wiki/File:Schema_of_the_layers_of_the_graphical_user_interface.svg


https://www.oreilly.com/library/view/linux-device-drivers/0596000081/ch01s02.html