

Kernel vs SO

udost. abstrakcje
zarządza hardwarem

programy systemowe
• potrzebne do prawidłowego funkcjonowania systemu

programy użytkowe

Zasoby:

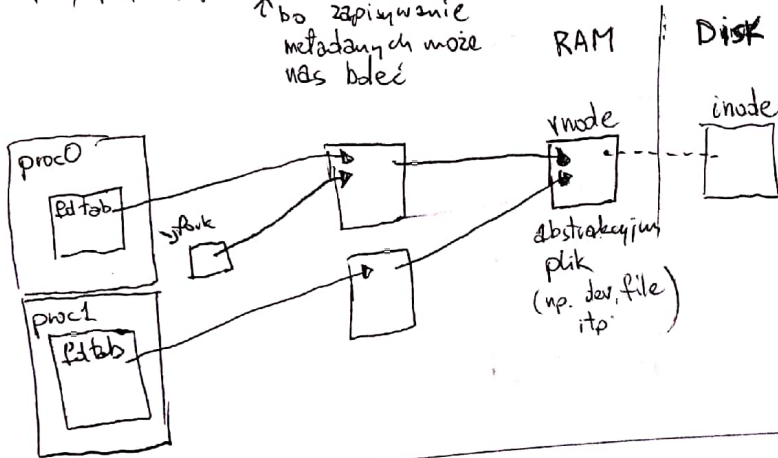
- czas procesora:
- warstwa abstrakcji do urządzeń
 - mapuje pamięć urządzeń do virt. address space

Izolacja i współdzielenie

- osobny address space
- niektóre rejony shared
- unveil (?)
- usernamespace (linux) groups →

Przechowywanie danych

- sync, fsync, fdatasync
- ↑ bo zapisywanie metodami może nas boleć



UPRAWNIENIA

- uwierzytelnianie - spr. tożsamości
- autoryzacja - spr. praw dostępu

szeregowanie zadań

policy - scheduler (planista) ← algorytm szeregowania
mechanizm dispatcher (dyspozytor)

SYSTEM WSADOWY

- dostaje ~~zadanie~~ listę zadań
- odpala / nadzoruje
- zwraca wyniki / (lub nie xd)

Process Control Block

→ wszystkie zasoby procesu
rejstry, VM

programowości kooperacyjna

→ yield (program sam oddaje władzę do procesora systemu)

wywołaszczowanie

- kwant czasu
- wsparcie sprzętowe (interrupt)

user space → kernel space

- ~~przekazywanie~~ przekazać na wykonywane instrukcje
- synchronizować cache wszystkich procesów
- ↳ bo każdy proces ma pamięć jądra i musi ją zaktualizować

kopiowanie pamięci z jądra/do jądra

- zawsze prosi o jądro
- jądro decyduje czy skopiować czy nie

Kod błędów

• errno

Punkt wywołaszczowania

strace - ślad wywołania

ltrace - ślad biblioteki

PROCESY

- rodzic - może się zmienić na inita lub innego reapera w przypadku śmierci rodzica
- priorytet - do szaregowania zadań
- proc filesystem
 - wszystkie dane procesów w postaci plików
 - można też gadać z kernelu
- interpreter: (#!)

PCB

- process state (ready, suspend)
- priority
- elapsed time (since suspend)
- ids of children
- IPC info
- pid
- Prog Counter
- Register
- privileges

- zasoby plikopodobne
 - fd zmapowane mmapem

- sesja - zbiór grup procesów na tym samym terminalu kontrolującym

Time

- usr w p. użytkownika (wszystkich wątków)
- sys w jądrze
- real od początku do końca wykonania
- $real\ systemie = usr + sys$

fork

- copy whole process (mark copy-on-write)
- copy fd-table

execve

- nowa przestrzeń adresowa
- maski sygnałów zostają

fork i pliki

- ten sam kursor lseek
- po close usuwa referencje

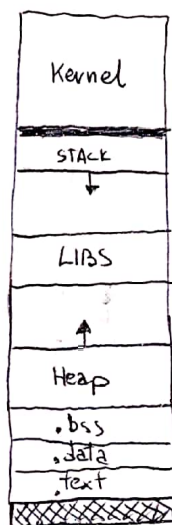
uid : real - id tego co odpala
effective
saved

set-uid → ustawia real

execve (name, args, envp)

environ - tablica stringów

Przeźreni Adresowa



Sygnat

SIGTUP
SIGTERM = SIGKILL

blocked → nie czeka

ignored → czeka ale nie dostaje

Session leader

"do obciążenia wszystkich
po zamknięciu terminala"

Grupy dodatkowe

abort - _exit() ale core dump
pause - czeka na SIGCONT

PLIKI

- sekwencyjny | swobodny dostep
↓
np sockety dev
↓
regular files

- ścieżka znormalizowana

→ bez kropki

→ z rozwiązanyymi symlinkami

- flagi: EXCL
ASYNC
SYMR
DSYN
TRUNC

- bloki
→ opomijany na
4kiB blokach
lub 52kiB

- mounting points
→ atrybuty

- gniazda → sieciowe
→ domeny unixowej

- SHORT COUNT → read/write

→ nie zwraca np. write gdy zabraknie miejsca na dysku
wtedy błąd a nie shortcount

- urządzenia → blokowe
→ znakowe

• katalogi



V - możesz przeczytać direnty

W - możesz robić wpisy

X - można wejść

sticky - usunąć może tylko właściciel/root

Inter Process Communication

strumieniowa 
datagramowa 

Gniazda domeny Unixowej
→ można wystać otwarte fd
(na swojej tożsamości)

porty do 1024 są
zarezerwowane

TCP vs UDP
connect() recvfrom()
accept() sendto()
listen()
bind()

netstat - informacje o połączeniach sieciowych

PAMIĘĆ

- błąd strony

minor - można ogarnąć (jest w RAMIE)
(np. podpiąć stronę nową)

major - ~~podpiąć~~ SEGV
trzeba ściągnąć z dysku

mprotect - zmienia uprawnienia obszaru pamięci

madvise - hinty dla mmapa
np. że będziemy używać sekwencyjnie

FRAGMENTACJA

- zewnętrzna

$\sum \text{free blocks} \geq \text{request}$ - ale nie możemy zaskładać

- wewnętrzna

~~payload < request~~

payload < request

WSPÓŁBIEŻNOŚĆ

- pula wątków - rozdzielamy zadania między istniejące wątki

owątki - osobny kontekst
- wspólna pamięć

mutex rekurencyjny

→ jeden wątek może wziąć ten sam

mutex

(przekazuje się przy funkcjach rekurencyjnych)

pthread_detach - sam się pochowa

Wątki użytkownika = coroutine

- nie są wspierane przez jądro

RPC - remote procedure call

skopane zadania

1.4

10.4

12.2,3