Operačné systémy

Linux

Ing. Martin Vojtko, PhD.

2024/2025

História

UNIX

- Prvá verzia UNIX vznikla na vyradenom PDP-7 v roku pána 1969.
  - Vývoj prebiehal na pôde Bell Labs Kenom Thompsonom.
  - pôvodne UNICS (UNiplexed Information and Computing Service)
  - vznikol prepísaním a osekaním MULTICS.
- UNIX PDP-11
  - Projekt pohltil celé oddelenie Bell Labs.
  - Presun z PDP-7 na PDP-11 podnietil vznik jazyka C.
    - Plán A (asembler) nemohol fungovať.
    - Najprv vznikol jazyk B vychádzal z neúspešného jazyka BCPL.
    - Dennis Ritchie preto vyrukoval s plánom C.
- 1974 publikovali Ritchie a Thompson svoju prácu o UNIX-e.
  - Univerzity prakticky okamžite začali UNIX používať.
  - V roku 1984 dostali za tento článok Turingovu cenu.
- Verzia 7 (1978) sa stala prvou portovateľnou verziou UNIX.

UNIX

UNIX distribúcie

- system V
- BSD -> FreeBSD
- XENIX
- MINIX -> Linux

POSIX (portable operating system standard)

Keďže UNIX bol open-source každý ho mohol prevziať upraviť, zabaliť a predávať pod svojou značkou. Takže nakoniec prvotná myšlienka jedného OS a prenositeľných programov prestala fungovať. IEEE preto zvolal zjazd všetkých distribútorov UNIX-u a rokovali o štandardizácií. Kupodivu to dopadlo dobre a vznikol skutočne fungujúci štandard.

Linux

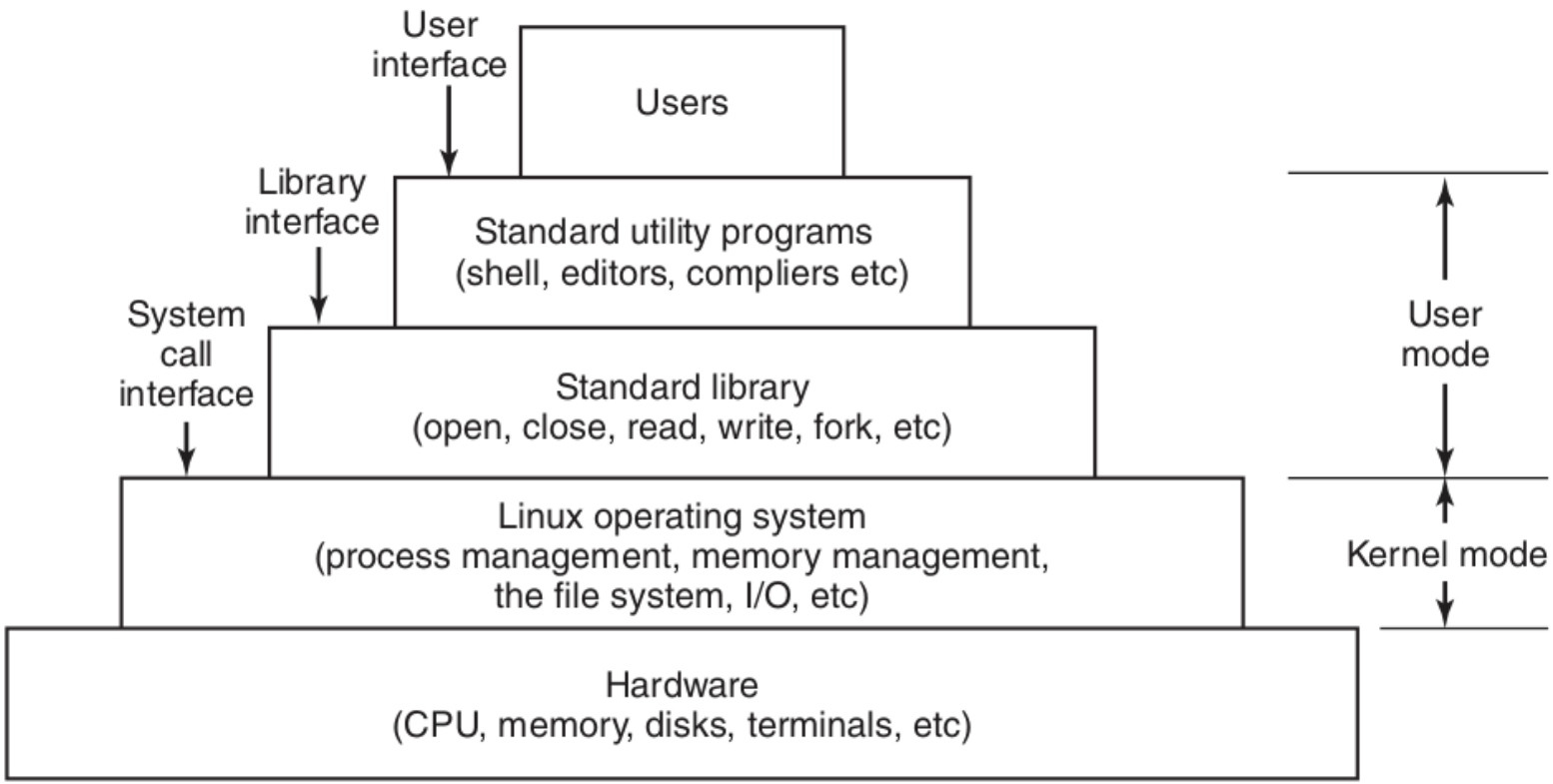
Linux

- Linux je licencovaný GPL (GNU Public Licence).
- To znamená, že každý môže použiť, kopírovať, modifikovať a re-distribuovať Linux za podmienky, že zverejní zdrojové kódy.
- Linux sa dostal na scénu vďaka súdnemu sporu medzi Berkeley a AT&T.
- FreeBSD (vyvíjaný od 1977) by bol silnou konkurenciou pre Linux (dva roky vývoja).

Linux - ciele

- jednoduchosť, elegantnosť a konzistentnosť.
- princíp najmenšieho prekvapenia.
- flexibilita
- jeden program robí jednu vec a robí ju dobre.
- odbúranie nadbytočnosti a duplicity.

Linux - rozhrania

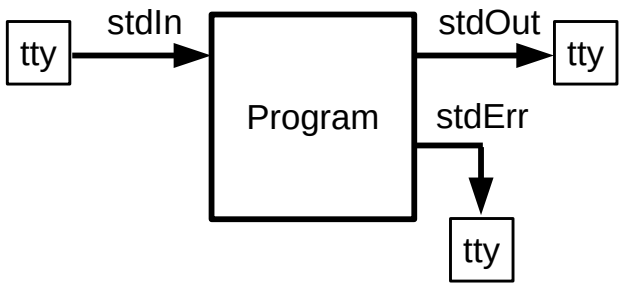


Linux - shell

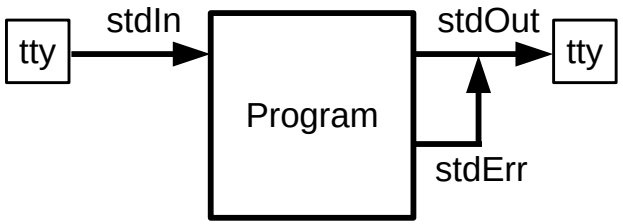
- shell je textové rozhranie určené na vykonávanie príkazov nad OS.
- po inicializácií shell zobrazí prompt.
- shell parsuje zadávaný text a identifikuje príkazy, ktoré má vykonať.
- príkaz/program môže pracovať s ďalšími parametrami a prepínačmi.
- shell a každý program v Linux-e má prístup k špeciálnym súborom:
  - stadard input - stdin 0
  - stadard output - stdout 1
  - stadard error - stderr 2

Linux - shell I/O

- program

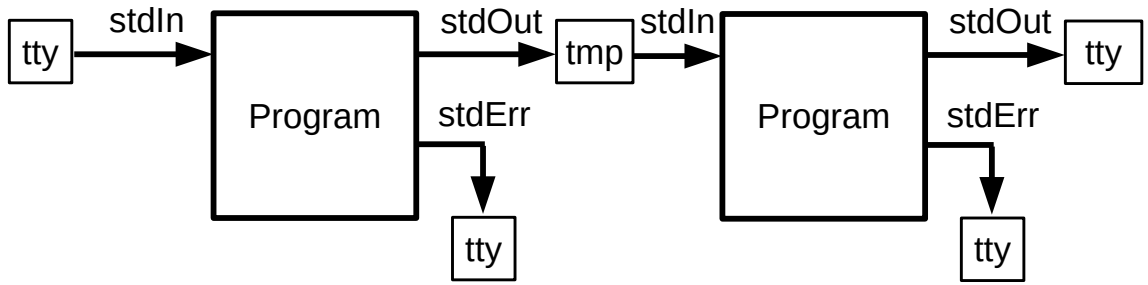


- program 2>&1

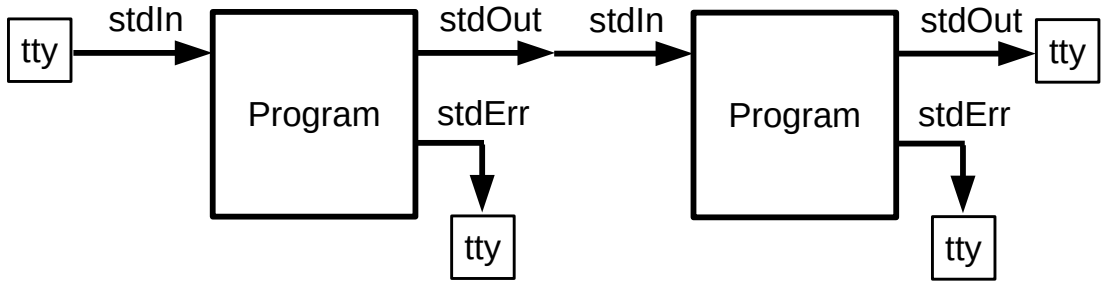


Linux - shell I/O

- program1 > tmp; program2 < tmp



- program1 | program2



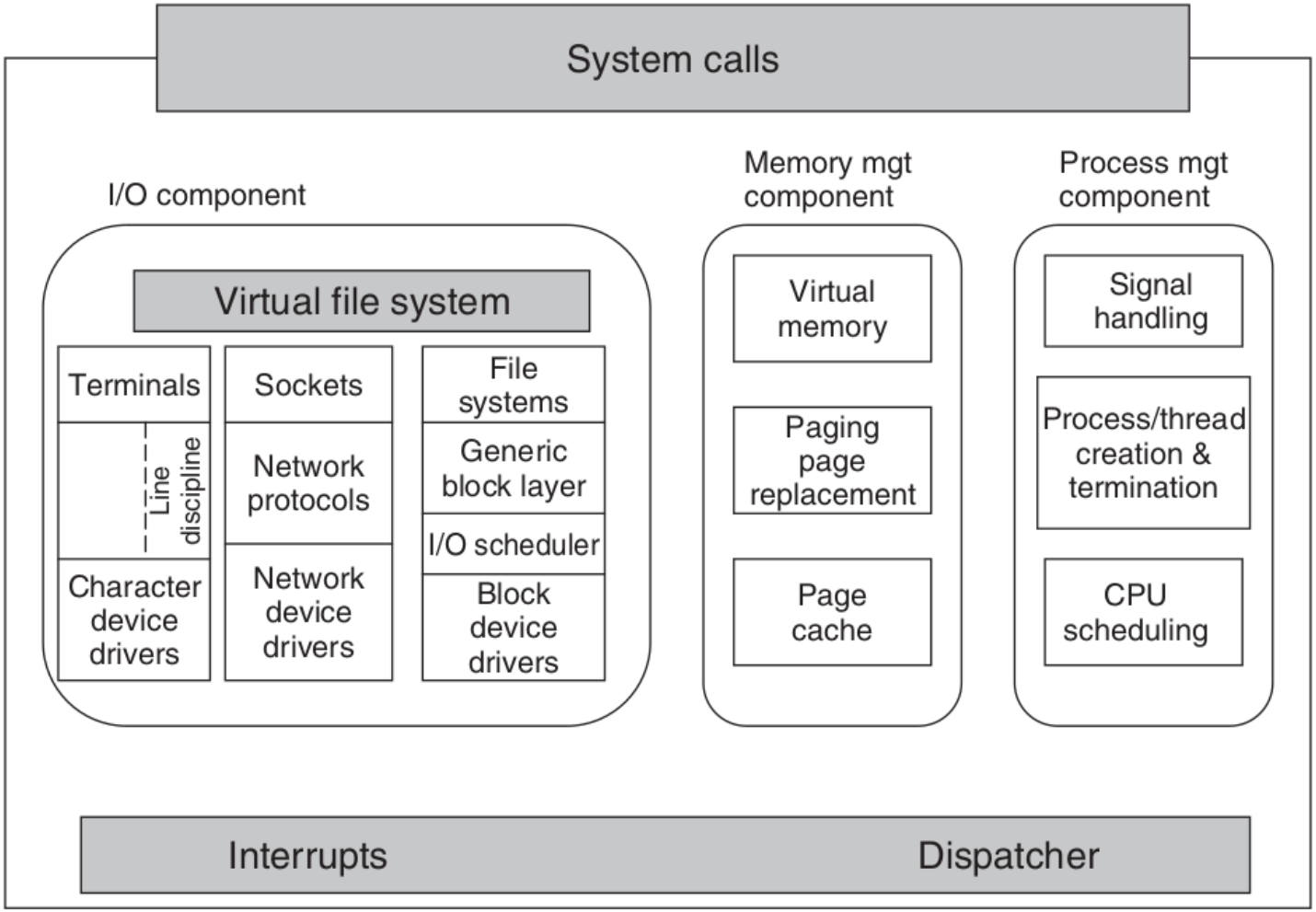
Linux - utilities

- POSIX 1003.1-2008 definuje 150 util programov pre Linux.
  - práca so súbormi a adresármi
  - filtre
  - editory a kompilátory
  - spracovanie textu
  - administrácia systému
  - rôzne
- Vďaka štandardu je možné napísať jednoducho prenositeľné skripty.

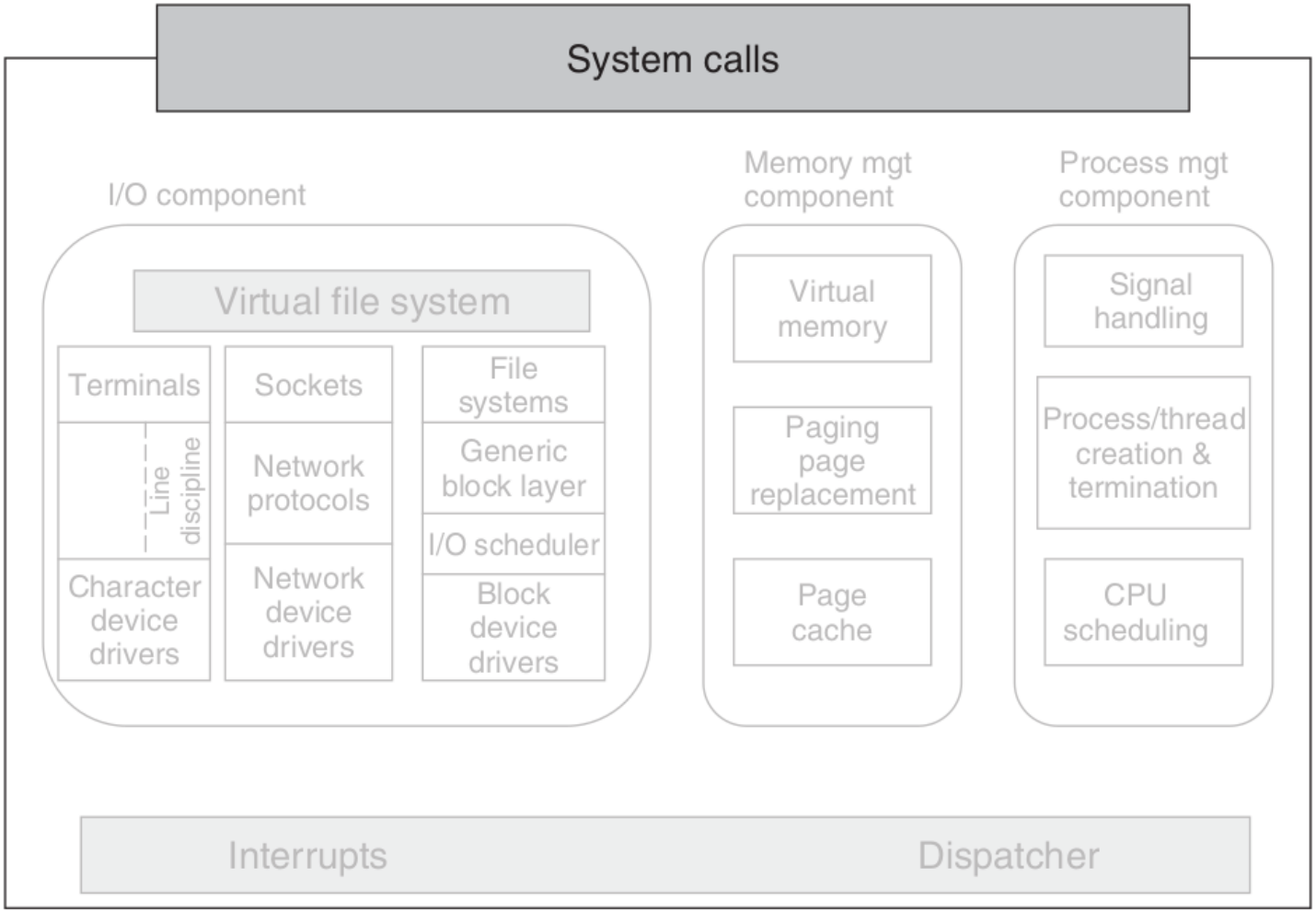
Linux - utilities

Program	Typical use
cat	Concatenate multiple files to standard output
chmod	Change file protection mode
cp	Copy one or more files
cut	Cut columns of text from a file
grep	Search a file for some pattern
head	Extract the first lines of a file
ls	List directory
make	Compile files to build a binary
mkdir	Make a directory
od	Octal dump a file
paste	Paste columns of text into a file
pr	Format a file for printing
ps	List running processes
rm	Remove one or more files
rmdir	Remove a directory
sort	Sort a file of lines alphabetically
tail	Extract the last lines of a file
tr	Translate between character sets

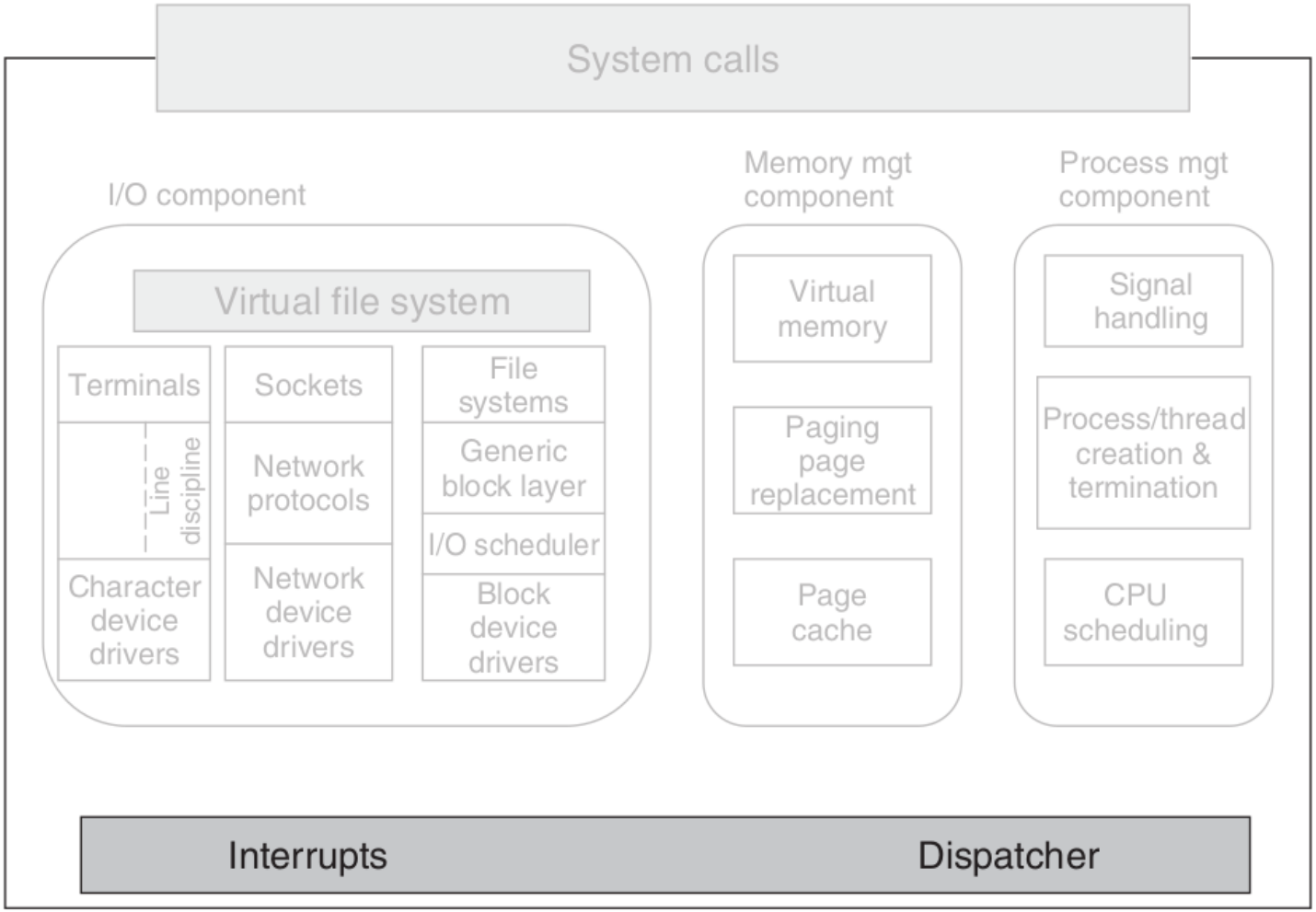
Linux - kernel



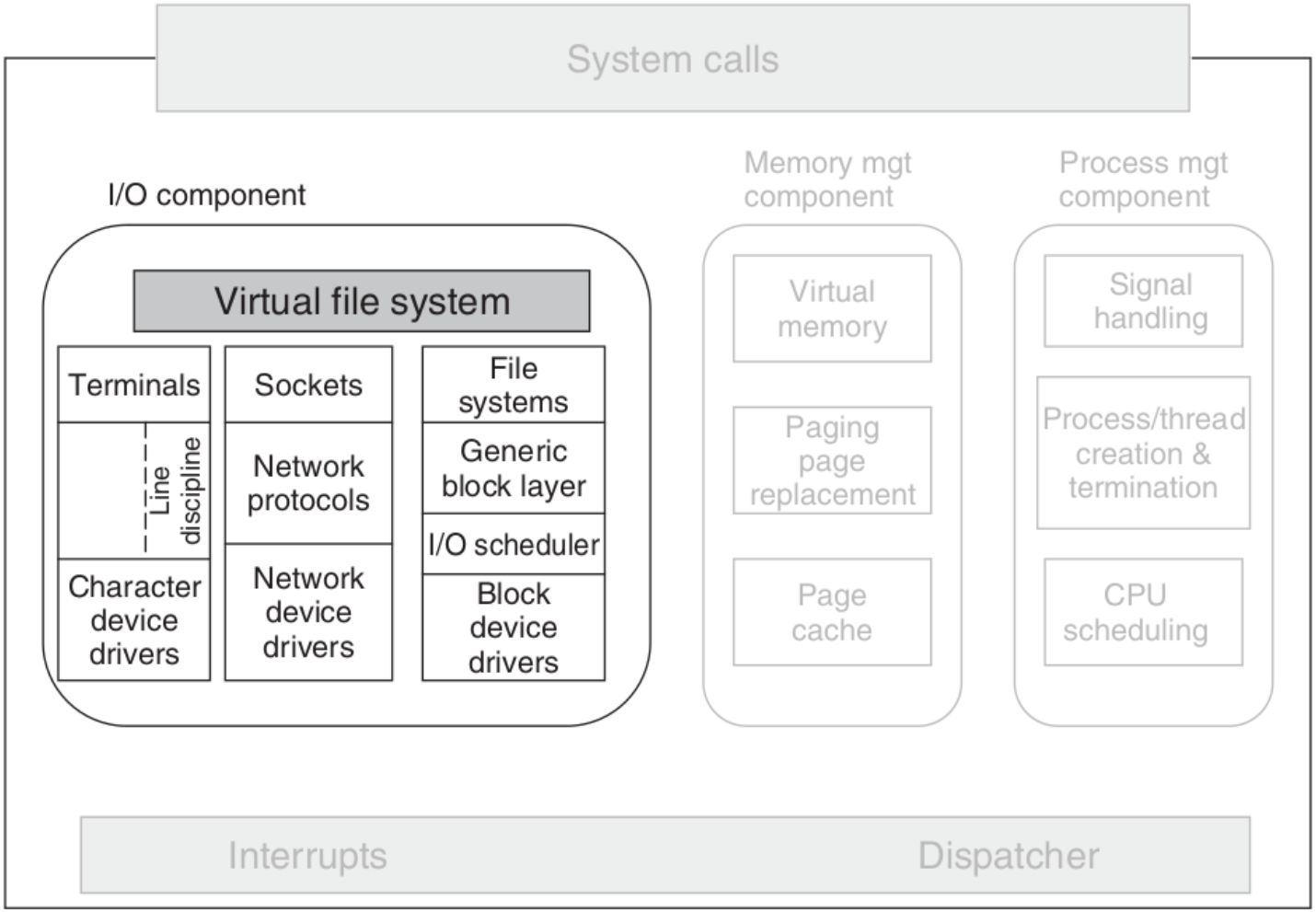
Linux - kernel



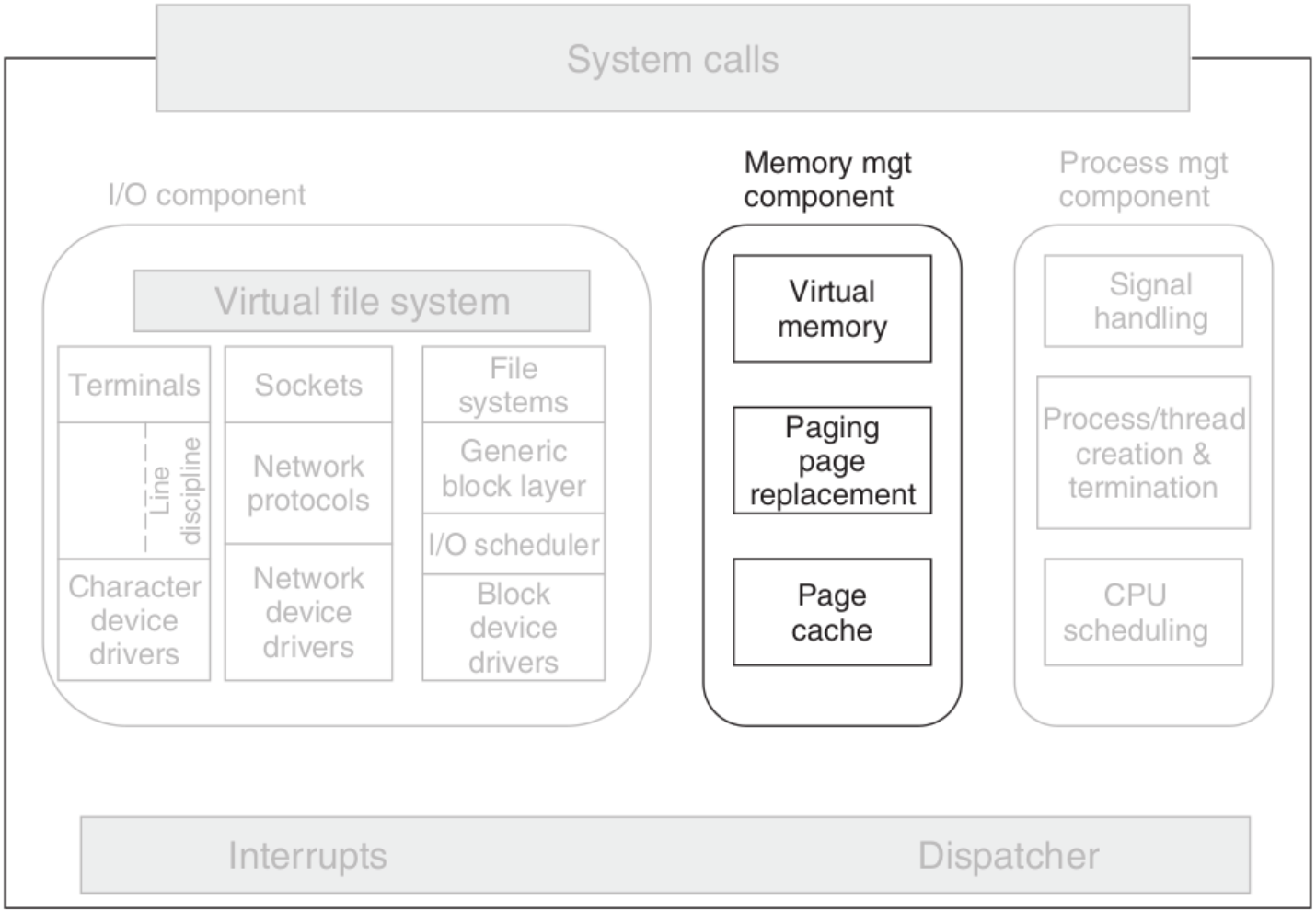
Linux - kernel



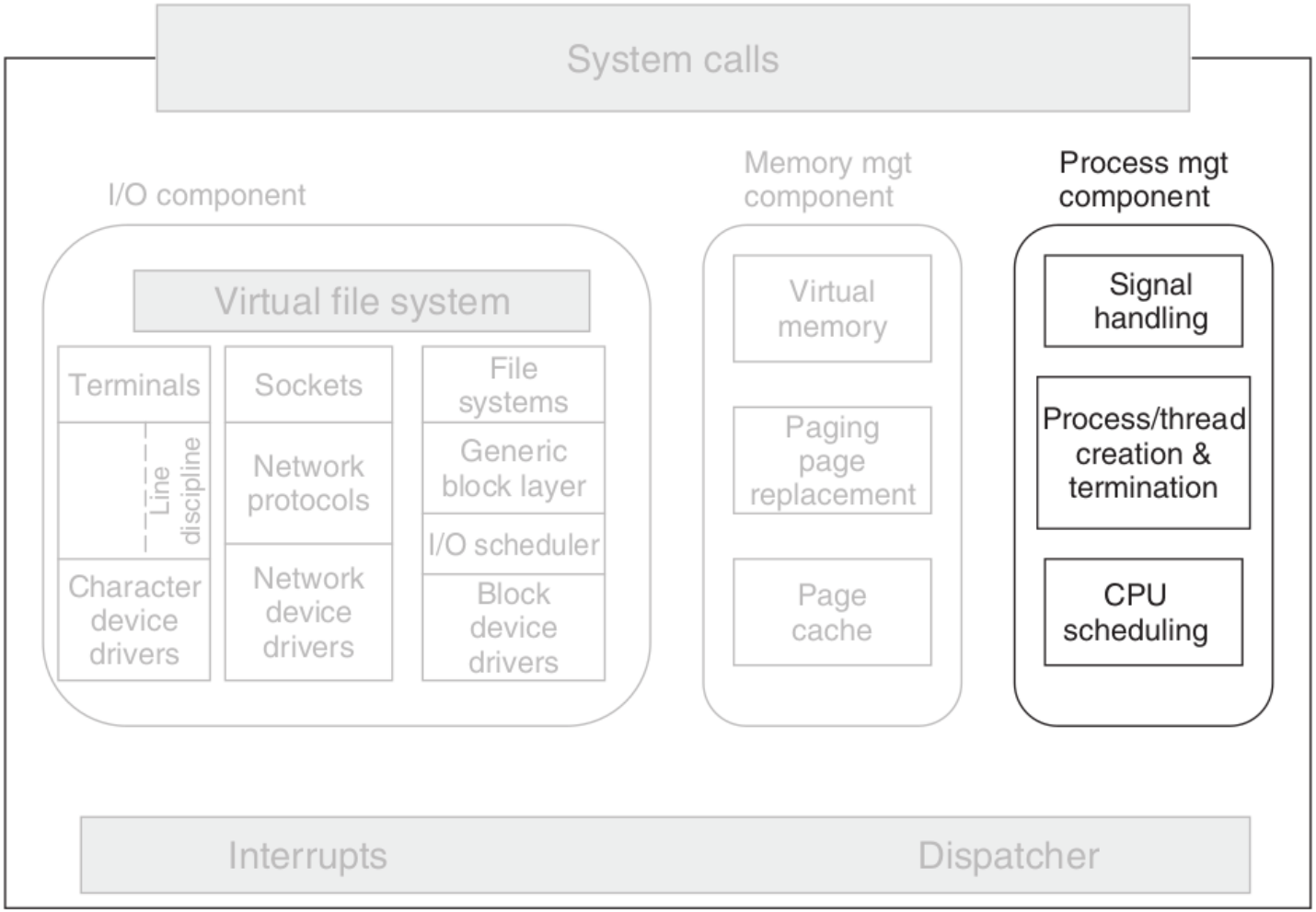
Linux - kernel



Linux - kernel



Linux - kernel



Procesy

Linux - procesy

- Proces spĺňa vlastnosti, ktoré sme už diskutovali.
  - vykonáva jeden program.
  - obsahuje jeden a viac thread-ov.
- Proces bežne spúšťa používateľ alebo systém.
- Linux podporuje aj daemon procesy bežiace na pozadí:
  - cron (od chronos - titan, personifikovaný čas) raz za minútu kontroluje či nemá vykonať prácu.
- Procesy navzájom tvoria vzťah rodič a dieťa.
  - Dieťa vzniká volaním fork(). (Niečo ako mitóza).
  - Dieťa dedí otvorené súbory ale nie spoločnú pamäť dát.
  - Rodič sa rozpozná tak, že fork() vráti PID dieťaťa.

Process - IPC

- pipe (rúra)
  - je najbežnejším spôsobom výmeny dát medzi procesmi.
  - umožňujú synchronizáciu pretože read() z prázdnej pipe uspí proces.
  - write() do plnej rúry uspí zapisujúci proces.
  - pipe je špeciálny druh súboru, ktorý sa bežne neukladá na disk.
- signál
  - procesy a používatelia môžu procesom posielať signály.
  - programátor môže definovať ako proces reaguje na signály.
    - ignorovať
    - zachytiť
    - ukončiť proces
  - proces môže poslať signál iba členom svojej process group.
    - rodičia a predkovia v priamej línii
    - všetci potomkovia

Process - POSIX signály

Signal	Cause
SIGABRT	Sent to abort a process and force a core dump
SIGALRM	The alarm clock has gone off
SIGFPE	A floating-point error has occurred (e.g., division by 0)
SIGHUP	The phone line the process was using has been hung up
SIGILL	The user has hit the DEL key to interrupt the process
SIGQUIT	The user has hit the key requesting a core dump
SIGKILL	Sent to kill a process (cannot be caught or ignored)
SIGPIPE	The process has written to a pipe which has no readers
SIGSEGV	The process has referenced an invalid memory address
SIGTERM	Used to request that a process terminate gracefully
SIGUSR1	Available for application-defined purposes
SIGUSR2	Available for application-defined purposes

Process - POSIX system calls

System call	Description
pid = fork( )	Create a child process identical to the parent
pid = waitpid(pid, &statloc, opts)	Wait for a child to terminate
s = execve(name, argv, envp)	Replace a process' core image
exit(status)	Terminate process execution and return status
s = sigaction(sig, &act, &oldact)	Define action to take on signals
s = sigreturn(&context)	Return from a signal
s = sigprocmask(how, &set, &old)	Examine or change the signal mask
s = sigpending(set)	Get the set of blocked signals
s = sigsuspend(sigmask)	Replace the signal mask and suspend the process
s = kill(pid, sig)	Send a signal to a process
residual = alarm(seconds)	Set the alarm clock
s = pause( )	Suspend the caller until the next signal

Process - Implementácia

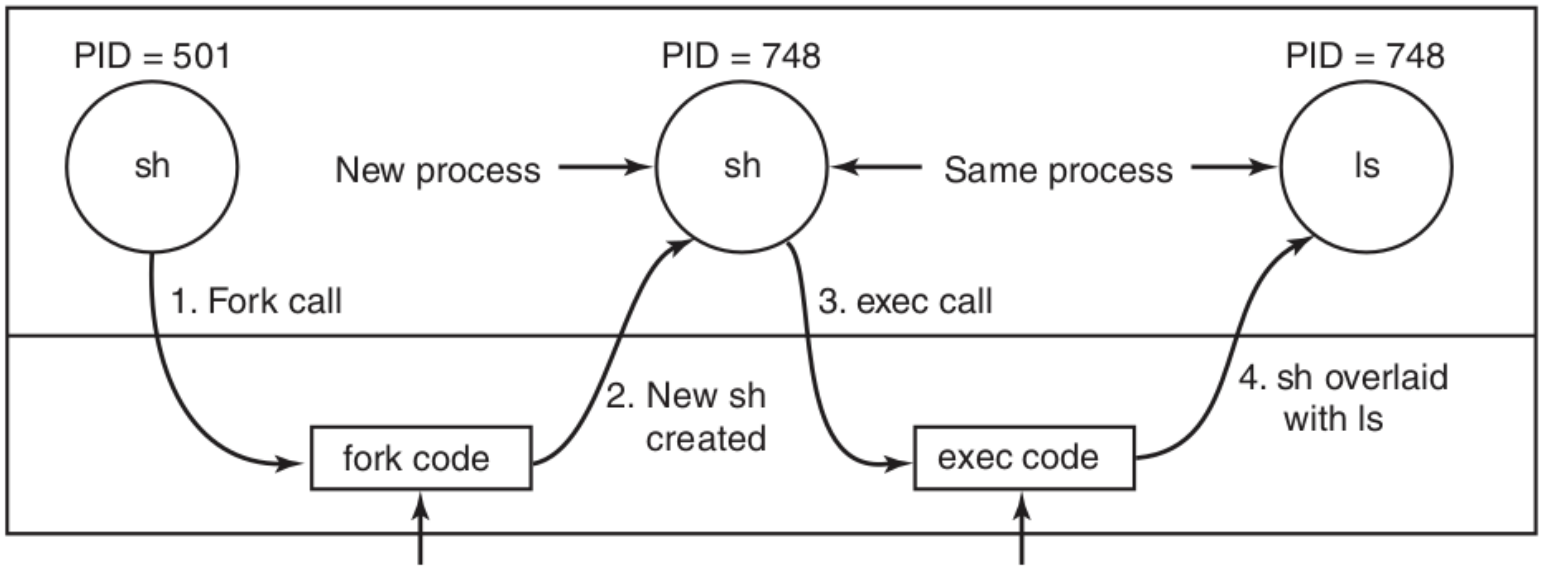
- proces je reprezentovaný ako množina bežiacich úloh, thread-ov.
  - proces má minimálne jeden thread
  - každý thread je opísaný v štruktúre task_struct.
- z pohľadu kernelu je každý thread braný ako proces.
- Kernel Linux-u je tiež viac vláknový - kernel threads
  - kernel thread nie je vlastnený žiadnym user procesom.
  - vykonávajú kód kerelu.
- procesy sú udržiavané v double-link liste. Pre okamžitý prístup je PID mapované na presnú adresu task_struct.

Process - task_struct

- Parametre plánovania: priorita, čas strávený na CPU, čas strávený spaním.
- Pamäť: **pointer** na text, data, zásobník a tabuľky stránok, pointer do swap.
- Signály: maska ignorovaných, odchytených, blokovaných a spracovaných signálov.
- Registre CPU
- Stav posledného systémového volania
- Tabuľka file deskriptorov
- Kernel zásobník

Process - fork

- zavolanie ls zo shell-u



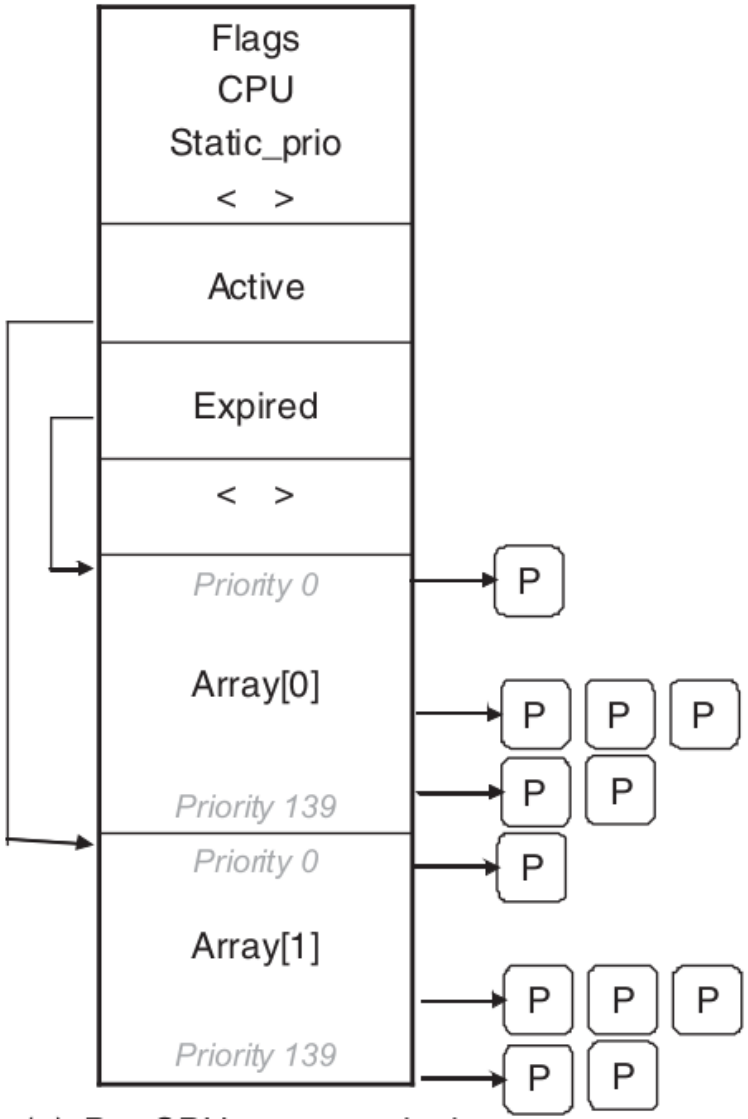
Plánovanie

- Linux plánuje thready nie procesy.
- Linux rozlišuje 3 triedy thread-ov.
  - RT FIFO - najvyššia priorita sú nepreeptívne 0-99
  - RT round-robin - rovnaké ako FIFO len sú preemptívne 0-99
  - timesharing - priorita 100-139
- Linux negarantuje RT napriek označeniu.
- počítanie času v jiffy (1000, 500, 250, 1HZ).
- Linux udržiava tzv. runqueue, ktorý obsahuje všetky ready úlohy.
- runqueue je udržiavaný pre každé CPU samostatne.

Plánovanie - plánovač O(1)

- operácie výberu a vloženia úlohy sú v zložitosti 1. Bez ohľadu na počet úloh v systéme.
- runqueue je organizovaný ako dve polia active a expired
- každé pole obsahuje 140 ukazovateľov na double linked list úloh s rovnakou prioritou.
- plánovač vyberá úlohu z najvyššej neprázdnej aktívnej úrovne.
- úloha, ktorá bola odplánovaná je vložená do active ak ešte má quantum. Inak ide do expired.
- ak nie sú žiadne úlohy v active poli plánovač urobí swap s expired. Takto sa zabezpečí, že úlohy s nízkou prioritou sa dostanú k slovu.
- priorita určuje veľkosť quanta (100 - 800ms, 139 5ms)
- snaha je všetky úlohy ukončiť čo najskôr.
- plánovač nevie dobre rozlíšiť interaktívne úlohy (users not happy)

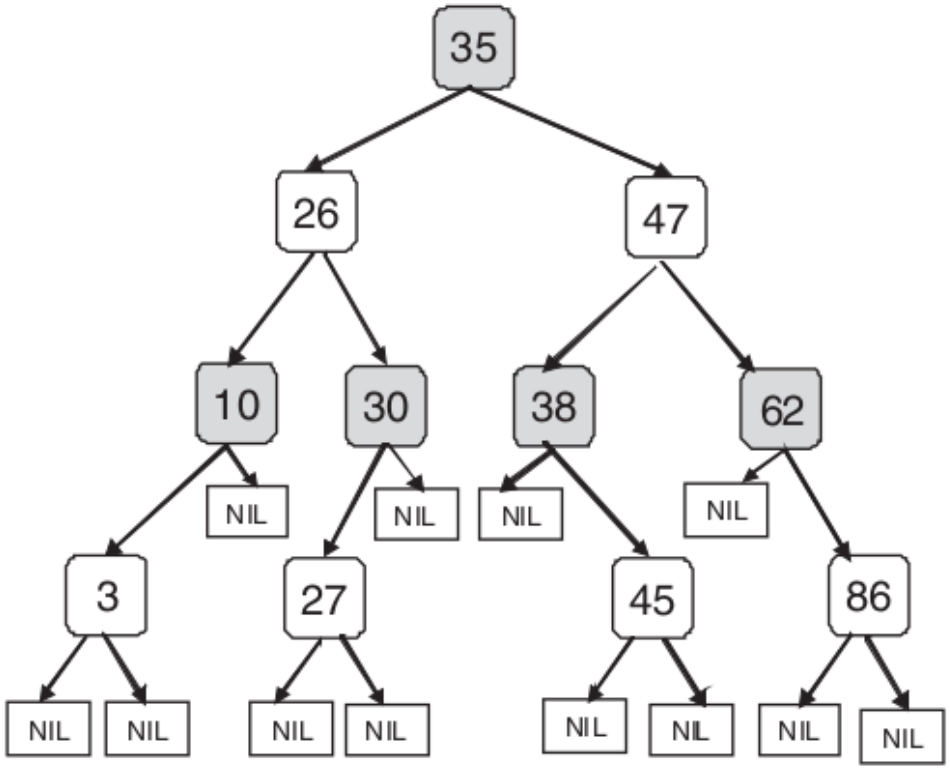
Plánovanie - plánovač O(1)



Plánovanie - plánovač CFS

- Completely Fair Scheduler. používa Red-Black tree.
- úlohy sú triedené podla času stráveného na CPU (vruntime) v ns škále.
- vnútorné vrcholy predstavujú úlohy vykonávané v systéme. Listy sa nepoužívajú.
- výber úlohy s najmenším vruntime je O(1)
- vloženie úlohy do stromu je O(log(N))

Plánovanie - plánovač CFS



Plánovanie - waitqueue

- waitqueue je objekt asociovaný s každou udalosťou, na ktorú môže úloha čakať.
- waitqueue obsahuje link list úloh, ktoré čakajú na udalosť a spinlock.
- spinlock chráni waitqueue pretože prístup do queue má kernel a interrupt handlers.

Synchronizácia

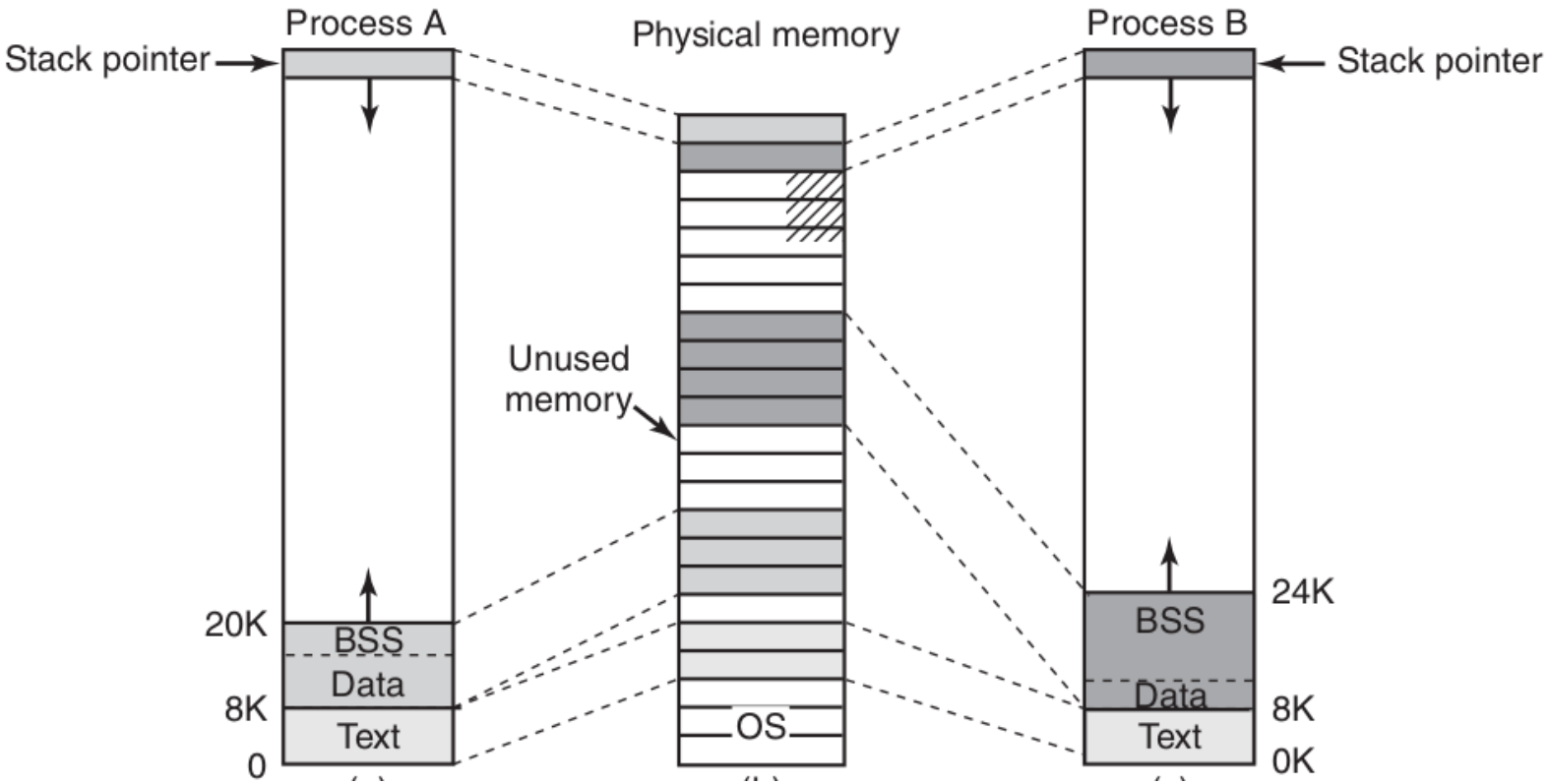
- Big Kernel lock
  - v minulosti celý kernel bol chránený takýmto zámkom.
  - prakticky akákoľvek práca s jadrom odstavila ostatné CPU.
- atomic_set a atomic_read - obálka okolo špeciálnych inštrukcií
- memory bariéry rmb wmb - zaručujúce vykonanie r/w operácií v poradí (a b c rmb/wrb d e)
- ticket alg. spinlock - pre úlohy, ktoré nechceme blokovať
- mutex a semafór - pre úlohy s blokovaním

Manažment pamäte

Manažment pamäte

- Model pamäte procesu v Linux-e je jeden z najstabilnejších modulov.
- Pamäť procesu pozostáva z troch segmentov text, data a stack.
- text segment je read-only. Jeho veľkosť a obsah sa nemení.
- data segment obsahuje všetky premenné programu, reťazce, polia.
  - inicializované dáta
  - BSS (Block started by symbol) globálne premenné nastavené na 0.
  - data segment môže rásť. (brk,malloc)
- stack segment - začína na konci virtuálneho adresného priestoru.
  - stack rastie smerom k nižším adresám.
  - ak presiahne 0. adresu je vyvolaný HW fault, ktorý pridá novú stránku do stack segmentu.
  - zásobník obsahuje po spustení procesu všetky vstupné premenné zo shell-u.

Manažment pamäte

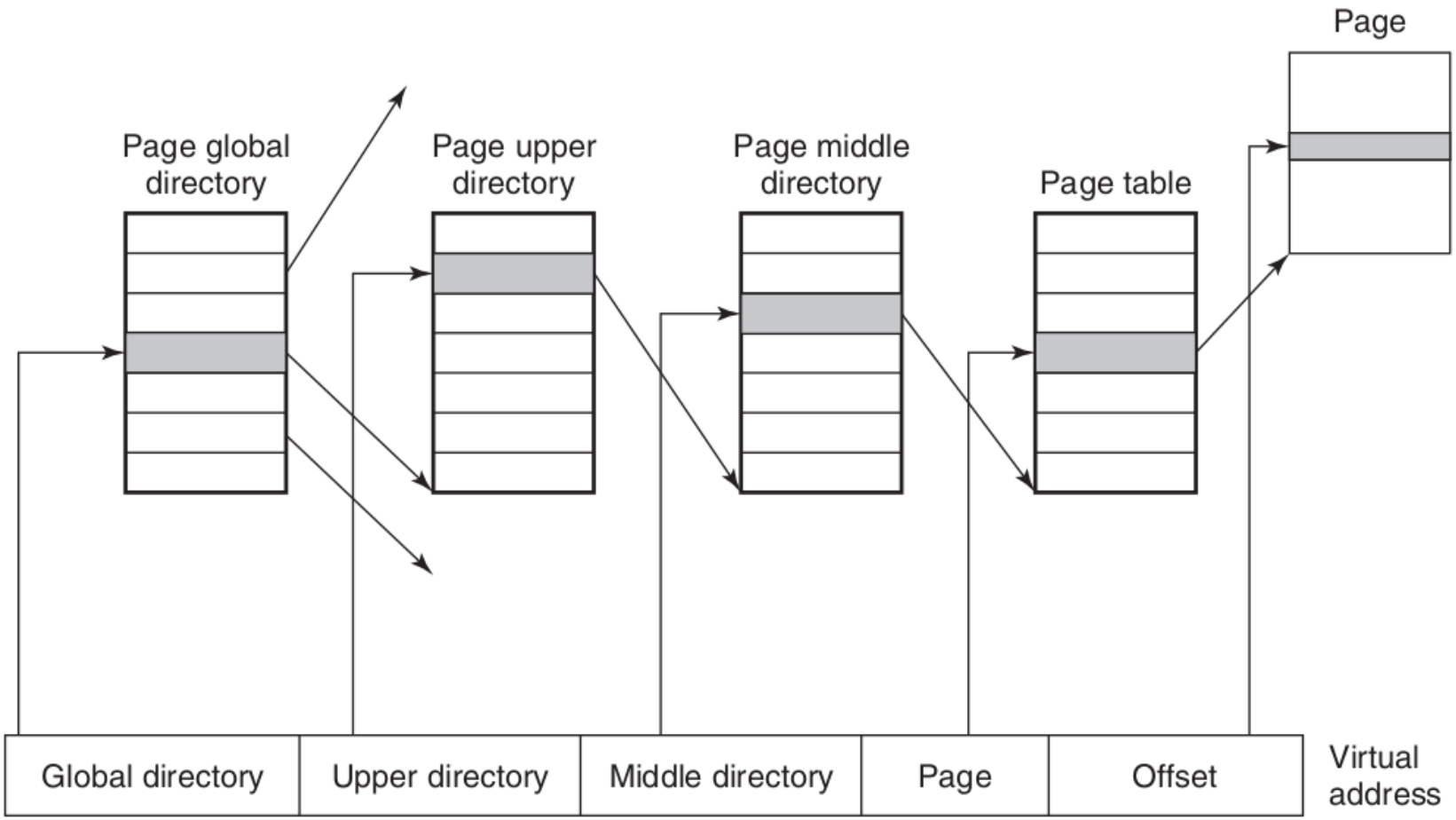


Manažment pamäte - implementácia

- 32b systémy user proces má maximálne 3GB. 1GB je vyhradený pre kernel.
- 64b systémy Linux podporuje maximálne 48b. 128TB je pre proces 128TB pre kernel.
- fyzická pamäť systému v Linux-e je rozdelená do troch zón:
  - ZONE_DMA ZONE_DMA32 - stránky systému, ktoré nemožno vyberať za obete plánovania. Nedajú sa ani odobrať procesu.
  - ZONE_NORMAL - normálne stránky použité pre bežné mapovanie stránok procesu.
  - ZONE_HIGHMEM - vyššie stránky, ktoré nemusia byť mapované permanentne. (i386 mapuje adresy od 896MB nepriamo)
- Pamäť Linuxu je možno rozdeliť na tri časti:
  - kernel a memory map sú pevne uložené v RAM.
  - zbytok pamäte je rozdelený do stránkových rámov.

Manažment pamäte - implementácia

- Linux používa 4-úrovňové stránkovanie.



Manažment pamäte - alokácia pamäte

- Page allocator - používa buddy algoritmus na prideľovanie voľných stránkových rámov.
- Slab allocator - používa buddy algoritmus na pridelenie veľkého úseku pamäte, ktorý následne kúskuje.
- Object cache - často používané štruktúry task_struct rovnakej veľkosti sú pred-alokované v cache. Cache s voľnými úsekmi odkazuje do vybraných Slab-ov.

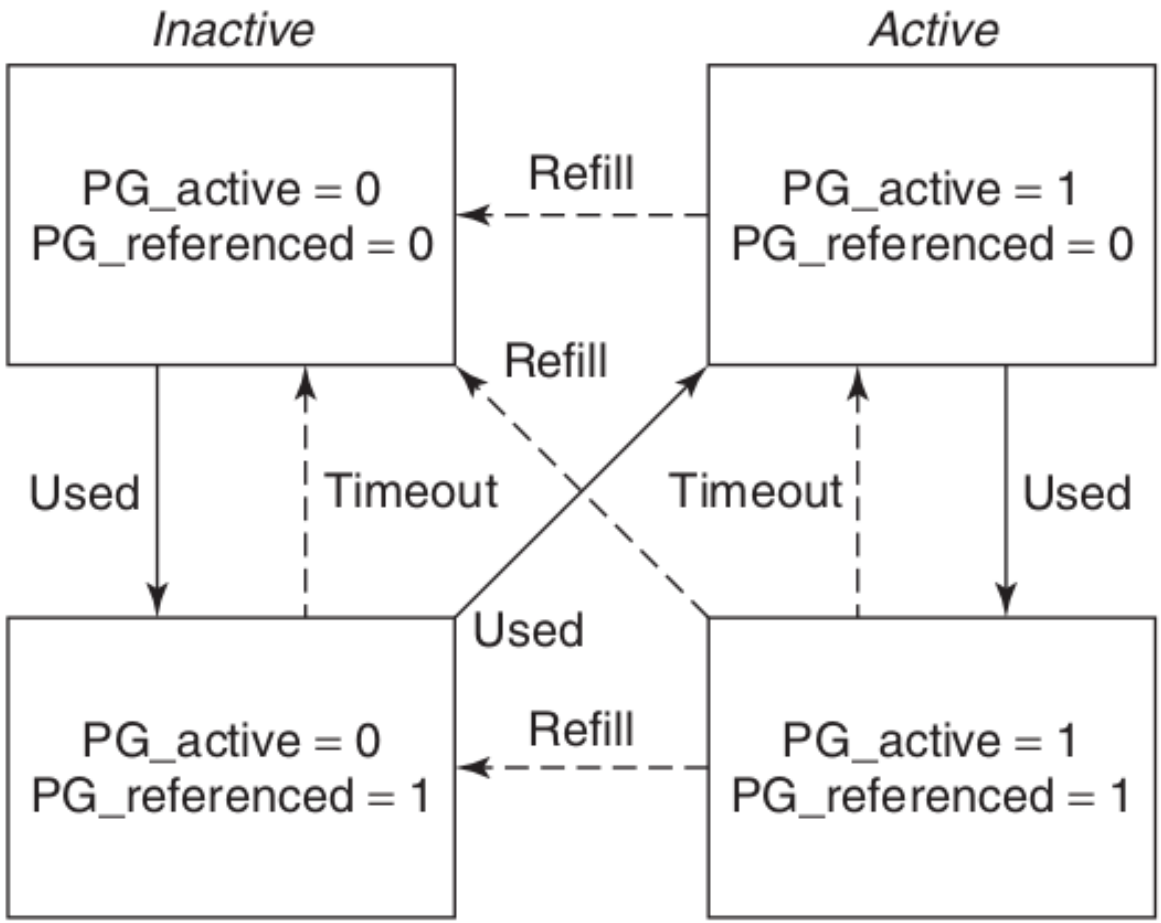
Manažment pamäte - paging

- Linux používa page daemon process na čistenie fyzickej pamäte.
  - proces sa periodicky spúšťa kontroluje stav pamäte.
  - ak je pamäte málo vyberá stránky, ktoré uloží na disk.
- Linux je demand-paging system. Nepoužíva pre-paging a ani working-set.
  - text segment a memory-mapped súbory odkazujú na konkrétne súbory na disku.
  - dáta a zásobník sú odkladané do swap-u - je partíciou na disku.

Manažment pamäte - PFRA

- PFRA - Page Frame Reclaiming Algorithm.
- Linux rozlišuje 4 typy stránok:
  - unreclaim-able - rezervované, blokované, zamknuté stránky
  - swap-able - stránky, ktoré je nutné zapísať do swap-u pred ich reclaim
  - sync-able - stránky, ktoré je nutné zapísať na disk (mem-mapped file)
  - discard-able - stránky na okamžité použitie
- kswapd - daemon v pravidelných intervaloch kontroluje stav pamäte:
  1. Discardable a nereferencované stránky.
  2. Čisté stránky (najprv nezdieľané a potom zdieľané)
  3. Dirty stránky označuje na zápis.
- pdflush - daemon zapisujúci dirty stránky na disk.

Manažment pamäte - PFRA

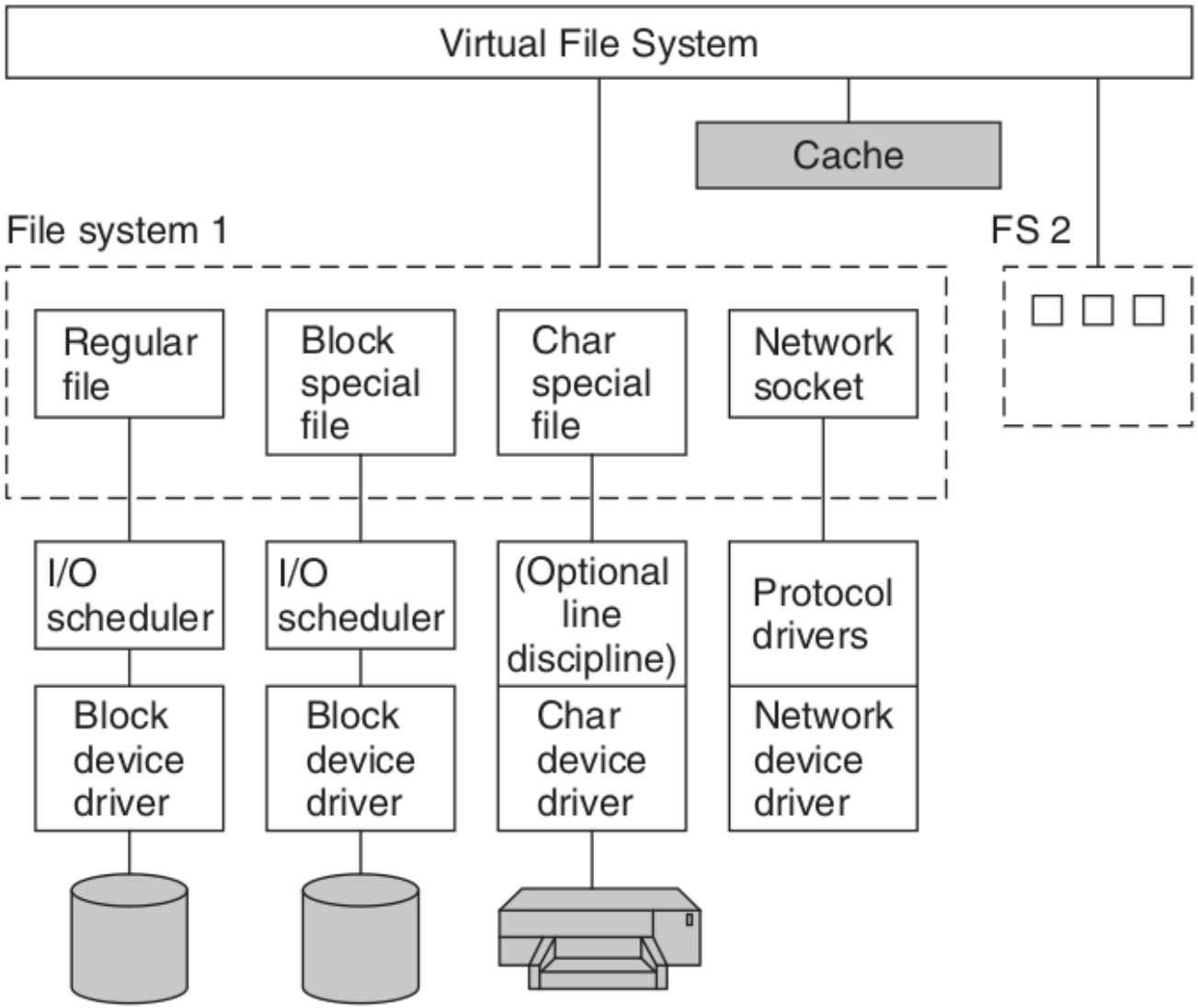


Manažment I/O

Manažment I/O zariadení

- Linux umožňuje prístup ku zariadeniam ich integráciou do FS.
- Zariadenia sú špeciálne súbory, ktoré sú identifikovateľné menom v adresári /dev.
  - block special files
  - character special files
  - network sockets
- Ľubovolné utility Linux-u s nimi pracujú ako s bežnými súbormi.
- Každý špeciálny súbor má pridelený device driver.
  - Major device number - identifikuje ovládač
  - Minor device number - sa používa ak ovládač podporuje viacero zariadení. Napríklad ak máme viac diskov tak každý má svoje minor number.
  - V špeciálnych prípadoch jeden ovládač ovláda dve zariadenia súčasne.
    - /dev/tty - ovláda klávesnicu a monitor.

Manažment I/O

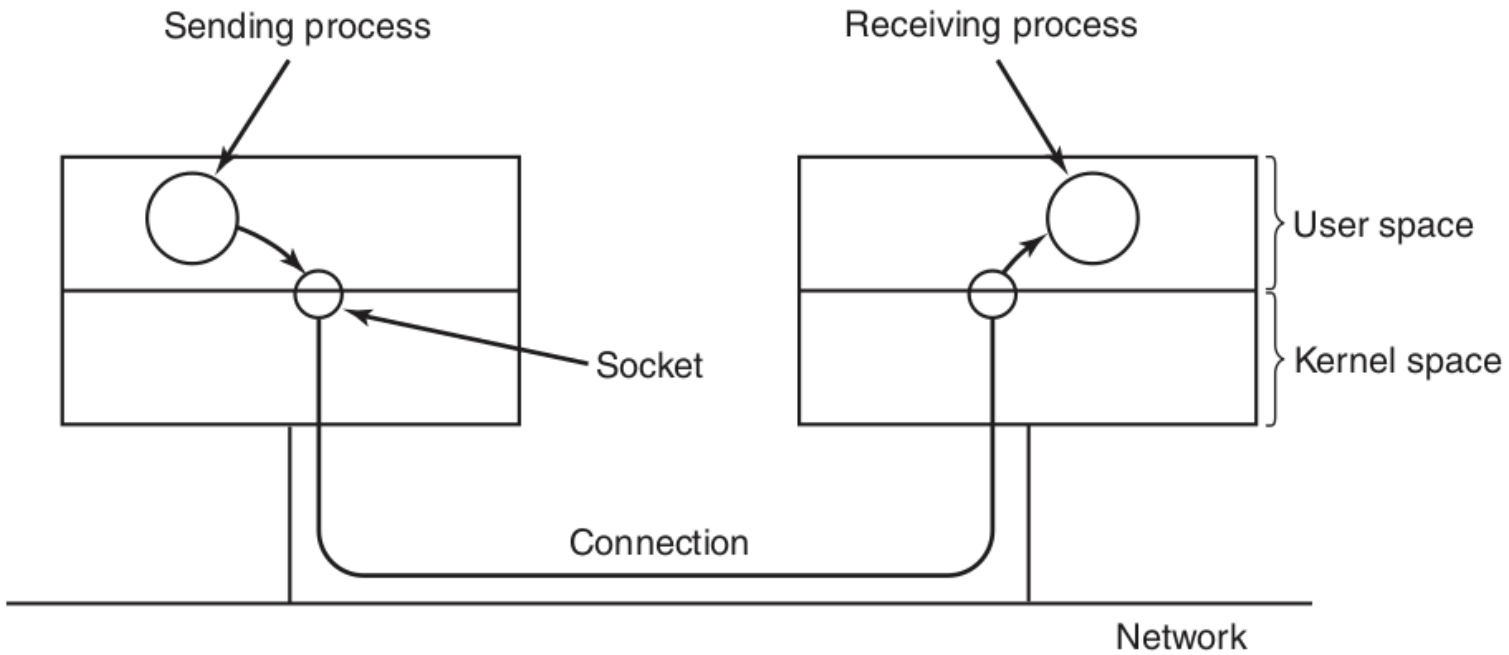


Manažment I/O

- block devices
  - I/O scheduler - usporiadanie požiadavok na sektory disku. (Linux Elevator Scheduler)
- character devices
  - Line discipline - odosielanie znakov na zariadenie až po spracovaní riadku.

Manažment I/O - network devices

- socket reprezentuje nadstavbu nad network ovládačom.
  - je to tiež súbor.
  - umožňuje definovanie typu prenosu TCP, UDP, raw.
  - aplikácia po jeho vytvorení dostane deskriptor.



Boot

Linux Boot

- BIOS (basic i/o system) - POST, sys a HW init.
- MBR (master boot record)- prvý sektor boot disku sa načíta do RAM.
  - mbr obsahuje malý loader, ktorý prečíta začiatok boot programu do pamäte.
  - boot program sám seba prečíta do RAM.
  - boot program prečíta root adresár z boot zariadenia. boot musí poznať FS.
  - boot prečíta OS kernel do RAM a skočí doň.

Linux Boot

- Kernel startup
  - startup napísaný v asembléri
  - inicializuje kernel zásobník
  - identifikuje CPU, veľkosť RAM, zakáže prerušenia, inicializuje MMU.
  - vola main() funkciu kernelu.
- Kernel C startup
  - inicializia štruktúr OS, a kernel page table.
  - čítanie auto-config súborov inicializácia zariadení.
  - živé zariadenia sa ukladajú do zoznamu s odkazom na ovládač.
- Kernel init
  - proces 0 - inicializuje rtc (rt clock), mount root fs, create init proces a page daemon proces.
  - init (proces 1) - koreňový proces pre ostatné procesy.
    - Pre single user mode vytvorí shell proces
    - Pre multi user inicializuje terminály pre prístup do systému (spustí x ďalších kópií).
    - Každý terminál spustí getty požadujúci login:.
    - Ak getty dostane login spustí login proces.
    - Ak login dostane správne heslo spustí shell.

Zhrnutie

Zhrnutie

- Linux je licencovaný GPL
  - jeho zdrojové kódy sú verejne dostupné a pod dohľadom.
  - do Linux-u prispievajú jednotlivci a aj organizácie.
  - organizácie platia členské poplatky do linux foundation.
- Linux je vyzretý a stabilný systém.
- Umožňuje efektívnu prácu skúseným používateľom.
  - preto je častou voľbou pre programátorov.
  - široko nasadzovaný ako serverové riešenie.
- Súčasné grafické rozhrania ho otvárajú aj menej skúseným používateľom.

Čo robiť do skúšky

- Prednášky, semináre a vaše data si nezabudnite skopírovať.
- Prelúskať materiály k predmetu:
  - prednášky - 1-11
  - cvičenia - príklady z 10-12
  - literatúra - Tannenbaum to čo sme prešli.
- Skúška v AIS forma textovej odpovede:
  - teoretické otázky - vymenujte, napíšte, opíšte, uveďte, ukážte, navrhnite
  - výpočet príkladov - **výpočet**, analýza, výsledok