

Tema 1

1) Описете как ОС разделят ресурсите на изчислителната система, дайте примери за основните типове разделяне:

a) Разделяне на пространството (памет) статично и непрекъснато-разпределение динамично и поблочно

Разделение на времето (процесори, други Ува)
DC трябва ефективно да управлява ресурсите на нашата, като
ги разделя между много програми и потребители, състезаващи се
за правото да ги използват.
Например, ако няколко процеса се опитат да използват едно и също
услуга, DC трябва да осигури използването на този

а) Ресурсът е част от него. Проблем, който не трябва да има част от част. Да осигури защита на частите. Справедливо да разделят.

б) Оперативна памет, чистосъвместима.

в) Процесите използват ресурса един след друг последователно. Задача на ОС: кой да е следващият? Колко дълго да се използва? Да обнема ли наследствено представения ресурс? Когато ОС наследства отнема ресурс от процес, издава, че ита преразпределение.

- централен процесор - с преразпределение
- паметни устройства - без преразпределение.

- опишете с 1-2 изречения работата на следните системни
издигвания в стандарта PCIEx:

извикваня в стандарта `fopen`.
Функцията `pipe(int pipefd[2])` - създава комуникационна тръба за

-pipe() pipe(int pipefd[2]) - създава външна
междупроцесорна комуникация (съвръга и приема).
Масивът pipefd с използване за връщане на 2 файлови дескриптора,
относещи се до краишата на трбдата:

относува и до краищата на пръстените сърцевидни бъбре на канал

- pipefd[0-3]-сочн към входа на канала
pipefd[4-7] същ към изхода на канала

- pipefd(1) или ком ~~назад~~ переход на канал.

- dup2(1) dup2(oldfd, newfd); Управи ф.г. newfd га ворува oldfd, ако newfd е врен
негово ворува нервният аргумент oldfd, ако няма това newfd е врен
нервно, то затваря.

-Фокусът дава нов процес-дете. Новостта дава един от процеса е постигнат идентичен на близките (този, който ѝ е създад). Той участва във всичките промени и отворява файлове и всички данни, свързани със стария родител. Те се различават по РДР (различни FP).

Те се разпределят по PID(process ID).
fork() връща -1 при неуспешно изграждане на процесите
за промените

exec() - изпълнява за процеса на програмата, която едн. процес изпълнява.

-wait()
pid_t wait(int *status); - процесът-родител чака своято дете да
приключи. wait() връща рид на завършилия процес-дете, а
status параметъра му дава кога на завършващия процеса-дете.
Процесът-родител чака завършването на процеса-дете.

-Waitpid()
pid_t waitpid(pid_t pid, int *status, int options); - укажваме на родителя кои
специфични процес-дете да чака. Чрез pid укажваме процесът-дете,
status-код на завършващия процес-дете, options-може да се
предотврати блокирането на родителя. Връща рид на завършилия
процес-дете.

Тема 2

1) Опишете разликата между времеделение и многозадачност.

• Какви ресурси разделя еднозадачна, еднопотребителска OS?

Многозадачност: може да се използва само 1 процес в дадено време.
Единственият потребител може да използва всички ресурси на системата.
Когато задачата е свързана с голем обем входно-изходни
операции, процесорът се използва много неефективно.
С цел по-ефективно използване на процесора се преминава
към многозадачна обработка: в оперативната памет се
намират едновременно няколко задачи и когато някое от
заданиета чака за изпълнение на входно-изходна

операция, процесорът може да премине към обработване на
друго задание. Основна разлика - при времеделението се позволява много
потребители да споделят системните ресурси едновременно,
докато многозадачността позволява на системата да

изпълнява няколко процеса/задачи едновременно.

Времеделение:

Времеделението е многозадачна работа, при която всеки потребител
има терминал. Времето на процесора се разделя между
всички потребители. При наличие на един процесор е
необходимо да се работи в реални на времеделение, които
нарежда на малки интервали от време, в които да се
изпълняват отединните процеси, превключвани чрез таймера,

2) Опишете с по 1-2 изречения работата на следните системни
извиквания в стандарта POSIX:

- `open()` `int open(const char *filename, int oflag, mode_t mode);` - откроява файл с име filename
- `open()` `int open(const char *filename, int oflag, mode_t mode);` - откроява файл с име filename
режим oflag, а ако не съществува, то създава с режим mode. Връща файлов дескриптор с който
се работи в програмата или -1 при неуспех.

- `close()` `int close(int fd);` - затваря файл с даден файлов дескриптор.

- `read()` `ssize_t read(int fd, void *buf, size_t nbytes);` - четене от файл с даден файлов
дескриптор, записва данните в buf (указател към областта от програмата, където се
записват данните), nbytes - колко байта са за четене. Връща брой на реално
прочетените байтове, иначе връща -1. (При край на файл - 0).

- `write()` `ssize_t write(int fd, void *buf, size_t nbytes);` - писане във файл с даден
файлов дескриптор, записват с nbytes байта от buf във файла с ф.г. fd.
Връща брой на реално записаните байтове, иначе връща -1.

- `lseek()`
`off_t lseek(int fd, off_t offset, int whence);` - използва се за позициониране
във файл.

whence-указва как се интерпретира откъсването

SEEK-SET - премини offset байта от начало.
SEEK-CUR - от curr_pos offset байти
SEEK-END - от края на файла до offset байти (напр.).

1

Тема 3

Дайте кратко определение за:

- многоделчна ОС - може да изпълнява повече от една програма в един момент.
- многопотребителска ОС - позволява на много потребители едновременно да имат достъп до компютора
- времеделение

Опишете разликата между многопотребителска и многоделчна работа.

Какви качества на ОС характеризират тези две концепции?

→ При многопотребителската ОС се позволява на няколко потребителя да изпълняват програми по едно и също време, докато многоделчната позволява на системата да изпълнява няколко процеса/задачи едновременно.

Качества: многопотребителска работа, разпределена обработка (управление на ресурсите), мултитърограмна работа.

2) Опишете с по 1-2 изречения работата на следните системни извиквания в стандарта POSIX:

- open() int open(char *filename, int oflag, mode_t mode); - открива файл с име filename в режим oflag, а ако не съществува, то създава в режим mode. Връща файлов дескриптор, с който се работи в програмата или -1 при неуспех.

- close() int close(int fd); - затваря файл с даден файлов дескриптор.

- lseek() off_t lseek(int fd, off_t offset, int whence); - използва се за позициониране във файла.

whence - указва как се интерпретира отместването

SEEK_SET

SEEK_CUR

SEEK_END

- pipe() pipe(int pipefd[2]); - създава комуникационна тръба (свързва 2 процеса).

Пасивният pipefd се използва за връщане на 2 файлови дескриптора, отнасящи се до краишата на тръбата.

- pipefd[0] ще има като вход на канал.

- pipefd[1] ще има като изход на канал.

- dup2() dup2(oldfd, newfd); - прави ф.г. newfd да сочи на същото място, като сочи първият аргумент oldfd, а ако преди това newfd е сочен член, то затваря.

Tema 4

Тема 4 съревнование на ресурси (race condition),
опишете ситуацията и дадете пример.

дайте пример. (изучение за ресурси)-проблем, при който някои

Race condition (competition of what body occupies).

процеса изгнания. А-опашка от сбукдания

Пример:

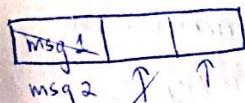
newer P

нравес Q

- (P1) $Q[e] \leftarrow msg^1_i$
 - (P2) $e \leftarrow e + \Delta_i$

- 9₁) $\theta[e] \leftarrow msg^2$; "изпраща".
 9₂) $e \leftarrow e + 1$; "напишката се увеличава."

31. В2. А. З. опашка



msg1 се започва и също така същите за избягване на race condition:

Опитете накратко инструментите за използване на ресурса:
Кодова секция, атмартна обработка на частни такава

(а) дефинирайте критична секция, ако може да настъпи такава ситуация, в която може да изпълнява изцяло, без да бъде прекъсвана от ОС.

прекъсвана от ОС.
⇒ Трябва да осигури критичната секция на задачи процес да бъде изпълнена като атомарна операция, т.е. Всички останали процеси трябва да изчакат завръщането на изпълнение от своята критична секция преди да влязат в собствените си.

преди да влязат в свободните сл. преди да влязат в свободните сл. преди да влязат в свободните сл.

(б) инструменти от нико ниво, специфични хардверни средства:

- enable/disable interrupt за еднопроцесорна система (взаимно изключване
- чрез изключване прекъсванията в кричичната секция).
- дополнителен бит lock (0-свободна, 1-заета структура), а метод

-spinlock (за многопроцесорна система)
- test and set - писане в паметта и Връщане старата стойност като
результат на операция.

- test and set - писане в паметта и отчитане на стойността
- atomic swap - атомарна инструкция за съдържанието на локална в паметта с синхронизация. Тя сравнява дадена стойност и само ако са идентични, модифицира съдържанието с новата стойност. Това е атомарна операция.

(в) инструменти от високо ниво, които блокират и събуфдат процес:

- семафори

2) Каква е спецификата на файловете в следните директории в Linux:

→ /etc - ^{настройки на конкретна OS} информация за потребителите (как се логва, какви са права, коя е /home директорията);

- структурата на файловете тук е варна за системните администратори.

- тук е записано и кои дискове се монтират при първоначалното зареждане, като можем и ние да добавяме такива дискове за монтиране.

→ /dev - не се съдържат файлове; описват се самите устройства, които са част от изпълнителната система; монтират се пъкът файлове, които имат

уникална структура и те задават какъв хардуер могат да обслужват и какъв реално обслужват; тук има логически изображения, които

предоставят драйверите на устройствата;

→ /var - ^{дани, речи за потребителите} съдържа разделени данни (примерно ако използваме бази данни тя ще се разположи тук); не е достъпна за обикновените потребители;

→ /boot - неща, необходими за зареждането на OS преди тя да влезе в нормално състояние;

→ /usr/bin - всички програми, които са вече изпълнени процеси;

→ /home - (home директория) - първото място, където се намирате след влизане в Linux системата; съдържа файловете на потребителя.

→ /usr/lib - включва обектни файлове (включвачи код) и библиотеки.

На някои системи може да включва и вътрешни двоични файлове, които не са предназначени за директно изпълнение от потребител.

→ /var/log

- основни файлове за регистрация

- файлове за достъп и удостоверяване (автентификация)

- файлове за инсталация и демонстриране

- системни файлове

- приложения

Тема 5

1) Хардуерни инструменти за защита (lock) на ресурса:

(a) enable/disable interrupt - за единопроцесорни системи (при еднопроцесорните системи се постига взаимно изключване просто като се изключат прекъсванията в критичния участък); те са машинни инструкции за забрана за рекурсивни функции, включващи spinlock.

(b) test and set
test and set(lock) - използва се, за да се пише в паметта и да се върне старата стойност като атомарна операция. Тя прочита **lock** и го записва в регистър

(c) atomic swap - атомарната инструкция се използва, за да се получи синхронизация. Тя сравнява съдържанието на локација в паметта с дадена стойност и само ако са същите, модифицира съдържанието с новата стойност. Това е направено като единична атомарна операция.

Опитете инструмента **spinlock**, неговите предимства и недостатъци.

Spinlock е механизъм за синхронизация от ниско ниво и е оптимален метод за решаване на проблема с race condition при кратки критични секции.

При spinlock се губи производителност, докато останалите процеси чакат за достъп до критична секция.

Други проблеми:

→ Може да настапи безкраен цикъл при стартиране на подпроцес, който ще опитва да достъпи горния spinlock, защото lock-ят е 1.

→ Може да се получи заукачане на процесора при хардуерно прекъсване, последвано от повтарящо се извикване на процедурата spinlock.

→ Губеже на ефективност

• За избегване на тези проблеми в началото на критичната секция се забранява прекъсванията на процеса

2) Къде е сместяваната на файловете в следните директории в Linux:

→ /etc

→ /dev

→ /var

→ /proc - поддържа на /dev; съдържа статистическа информация за всичко - кои процеси са активни, времето между отделни процеси, връзки с интернет, пъевдофайлове;

→ /bin - важни програми, необходими за стартиране на OS

→ /sbin - главни файлове на системата, които са обичайни за всички потребители (ps, ls, ..)

→ /home

→ /usr/doc - централна директория за документация.

Документацията е разположена в /usr/share/doc и е свързана от тук

Тема 6

1) Описете понятията приспиване и събудване на процес (block/wake up).

Сенатор-дебитинг и реализация.

Семафор-дифитиция
Опишете разликата между силни и слаб семафор

Семафор-механизъм за синхронизация от Високо ниво
записване на семафор

Структури от данни, необходими за реализация на семафор са:

- Структури от данни, необходими за реализация на алгоритъм
- бројче сътърбът процеси, които могат да бъдат допуснати до ресурса
- контейнер L (инфо как процеси чакат достъп до ресурса)

Основни процеси:

- конструктор `init(int x)`
 - Метод `wait()` - при этом за
должен go ресурса
 - Метод `signal()` - завершаете
должен go ресурса.
(освобождение на ресурса);

```

s. initl(0)
cnt = cnt + 0
L = Ø
lock = 0

```

```

5. signal()
    spin-lock(lock)
    cnt = cnt + 1
    if cnt <= 0
        pid = L.get()
        wakeup(pid)
    spin-unlock(lock)

```

```

S.wait()
spin-lock(lock)
cnt = cnt - 1
if cnt < 0
    L.put(self)
    spin-unlock(lock)
    block()
else
    spin-unlock(lock)

```

Семафорът е спел, когато е реализиран чрез обикновена опашка - витали активирам процеса, блокиран тай-рато.

процеса, блокиран чак-ратно.
Семафорът е слаб, когато не е реализиран чрез обикновена очилка.

Linux:

устройства, именуя их в Idev-

- въвеждането на файлове
 - идеята за съборка на файлове
 - описват се самите устройства, които са част от изчислителната система;
 - напират се псевдо файлове, които имат уникална структура и те задават какъв хардвер могат да обслужват и какъв ъзанъ обслужват.
 - тук има логически изображения, които предоставят драйверите на устройствата.
 - псевдофайлове в BIOS - подобно на IDE
 - съборка логистическа информация за другото - кои процеси са активни, връзките между отделни процеси, връзка с интернет.
 - Търди линкове - запис на директорията, която асоциира име с файл във файловата система. Всички директории-съдържани файлови системи трябва да имат поне 1 търд линк, давайки оригиналното име за всеки файл.
 - символни линкове - прокор на всеки файл, който съдържа редференция към друг файл или директория във формата на абсолютен и отношен път.

- Команда `ln`
`ln [OPTION] TARGET [LINK-NAME]` - създава линк като спецификация TARGET с допълнителната настройка LINK-NAME. Ако LINK-NAME е пропуснат, линкът се създава със общото базово име като TARGET-а в текущата директория.
- Сокети-двуносочен комуникационен канал като не е необходимо обектите страни да са последници на обич родител.

Тема 7

1) Взаимно изключване - допускане само на един процес до общ ресурс.

Опишете решение чрез семафори.

Мутекс-гарантира, че само една нитка ще достигне споделената променлива.

Семафорът се инициализира с 1-когато Mutex=1 първата нитка, която се е добре до него ще изпълни секущата си, но втората <0 ще се блокира, докато първата достигната не изпрати сигнал.

2) Качества и свойства на конкретните файлови системи,

реализирани върху block devices.

Ефективна реализация, отлагане на записа, алгоритми на асансиора

Тема 8

1) Комуникационна тръба (pipe), която съхранява един пакет информация – реализирана чрез редуване на изпращач/получател.

pipe е буфер-тръба, съхраняваща н пакета информация.

Използване на семафорите като броачи на свободни ресурси.

2) Права и роли в UNIX

команда chmod – промяна на правата на достъп до даден файл

или директория.

Има 2 начин за смяна на права на достъп:

1) Зададен чрез числа в 8-рична бройна система: chmod 0644 * (-rw-r--r--)

2) Зададен чрез букви: ита следния синтаксис [какво]

Как – user, group, others, all (права)

Действие: + - дава/не дава право

- - премахва право

= - меня изцяло правата за достъп с тези от дясната страна на действие.

Какво: read, write, execute (роли)

chmod go+w

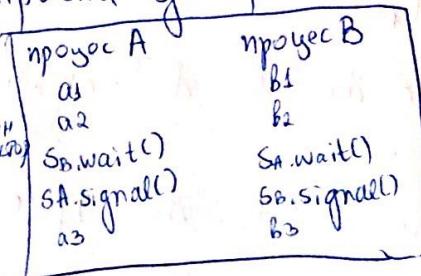
Тема 9

1)

Взаимно блокиране (deadlock) - проблем, възникващ при използване на семафори. Понякога се прилага и за взаимно блокиране на два процеса и няма кой да ги разбие. Това ще ги присни до края на изпълнението на изчислителния процес.

Възниква, когато:

- 1) процес изп. ресурс икономично
- 2) процес се нуждае от поне 2 ресурса
- 3) попадне на ресурс не може да се прекогрене
- 4) цикличност



Гладуване - проблем, който се появява, когато на процес не му бъде доставен достъп до нуждите ресурси, без които този процес никога няма да може да завърши своята работа.

Обикновено гладот е причинен от прилаганието на твърде опростен алгоритъм за съставяне на разписания, по които процесите да се изпълняват.

Пример: загара за философите и макароните
Има 5 философи на кръгла маса, които ядат макарони, но имат сервирани по един комплект прибори (отдясно). Ако всеки последен по едно и също време да яде първо към дясната вилча, то когато последне към лявата, все е злата от съседа.
Това ще доведе до липса на ресурси (нямат достъп до общу споделен ресурс), за да извършат процеса си. (deadlock)

2) Една паралакса файлова система в UNIX

Файлове и директории, команди - cd, mkdir, rmdir, cp, mv, rm
В UNIX файловата система е паралакса, държаваща. Съставена е от единство директории. Корентът на паралаксата е "/" (home).
Файл-основната единица, чрез която се осигурява съхраняването на постоянни обекти данни. Всеки файл има име (из от символични разширения).
Директория (каталог) - структура в ос, чрез която може да разделим отделни файлове, да ги обособим по някакъв критерий.

Файловете и директориите имат атрибути - размер, дата и време на създаване, достъп, собственик, права за достъп, пароли, флагови.

Типове файлове:

- обикновен файл (regular file) - съхраняване инфо върху външен носител
- специален файл (special file) - block (B) - върху - аBSTRACTИЯ за външно устройство
- каталог (справъднически директорий) - инфо за път или У файлове в системата.

- cd - сменяне директорията

cd arg

абсолютен/относителен път на директорията

cd без аргумент - върви на хъм пачаката потребителска директория.

- mkdir

mkdir да [дължина]-създава празни директории.

- rmdir

rmdir да [дължина]-трне само празни директории

- cp

cp [source] [destination] - копира файлове

И-вот файлове path-to-dir

cp f1 f2 - копира съдържанието на f1 във f2.

- mv

mv f1 dir1 - премества f1 в указаната директория

mv f1 f2 - премества f1 на f2.

- rm

rm f1 [f2...fn] - трне файлове

rm -r dir - рекурсивно трне на непразна директория.

Тема 10

1) Процеси в многозадачната система.

• Превключване, управлявано от синхронизация.

Превключване в системата с времеделение (time interrupt).

A) CPU bound (background) - извършват малко входно-изходни операции, но сметат много, т.е. трябва им много време сложено време (т.е. те са в Running).

B) I/O bound - процеси, които основно или често извършват входно-изходни (interactive, foreground/интерактивни, първите процеси операции правят малко сметки често са в Ready, а не в Running)

C) Realtime - приложения, които комуникират с околния свят.
имат deadline (краен срок) на време за извършване на процеса.

• ОС, използващи A и B (смартфони, сървери, лаптопи) - те са time sharing, но не гарантират realtime

• ОС, използващи C (ракети, коли)

Task Scheduler - това, което решава как да се превключва процесът (от чакане в работещ) - осуществлява се чрез хардуерния елемент time interrupt (превключване в системата с времеделение) - разделянето на времевите процеси; определя какъв тип е процесът и му слага приоритет.

2) Описете функционалността на следните команди в Linux:

• ls - без аргументи ни дава съдържанието на текущата директория (само имената на файловете).

ls dirs

Опции: -l; -a; -r; -t (време на създаване)

• who - показва информация за логнатите потребители

who -u - показва активните потребители в даден момент.

• find find [dir-
-t=be-searhed[arguments]] - търси файлове в иерархиата на директорията
-name, -type, -user, -group, -size, -exec

• ps - показва процеси на текущия логически терминал

ps -a (процеси на потребител)
-u (собственик на процес)
-x (сървъри с друг терминал)
Може да сортира процесите по CPU usage, memory usage и runtime.

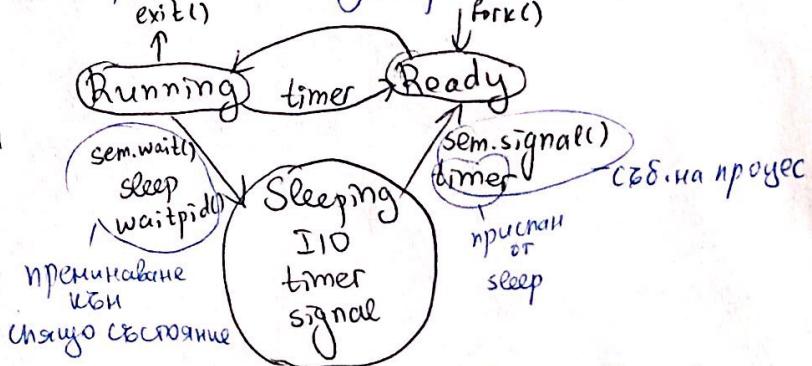
Тема 11

1) Възможни состояния на процес. Механизми и структури за приспиване/събудждане.
Фиаграна на состоянията и преходите между тях.

Състояния на процесите:

- running - имат работа за въртене, използват СРИ-то.
↳ Running - смятат нещо в момента; не чакат външна операция да настани.
- Ready - имат нужда от ресурс, но нямат достатъчно СРИ
- sleeping (S) - спящи процеси, които в момента нямат работа (нямат нужда от продължително време; имат нужда от изчислителен ресурс).
Спят, защото → очакват завършване на външна операция
→ очакват времеви момент (timer)
→ очакват сигнал от друг процес (signal)
- stopped
- zombie - заподобен спирателен (изход), но него е завършил, т.е. не е освободен от ресурси.

- събудждане на процес, прислан с waitpid() - когато някой друг процес изпълни своята команда exit().



Running $\xrightarrow{\text{timer}}$ Ready - спиране на сигнал (изтичане на предоставеното време)

Ready \rightarrow Running - ключов момент как работи ОС-частта, която взима решение как да превключва процесите и да сменя състоянието. Когато СРИ-то е освободи, то временно се използва от ОС (системно време). Когато трябва да вземе решение след като е бил спрян/прислан един процес кой да пусне, има различни схеми - спечелен алгоритъм (диспачър) взима това решение.

2) Опишете функционалността на следните команди в Linux:

- vi - текстов редактор; има 2 режима на работа $\begin{cases} \text{insertion mode} \\ \text{command mode} \end{cases}$
- tar - tape archive - команда за правене на архиви
- tar cf archive.tar file1 file2 - създава архив с име archive.tar, който съдържа file1 и file2. Има 2 вида компресиране на данни - със и без загуба на информация.
- Опции:
 - z - укаzia на tar да компресира данните, не просто да ги обедини в по-голям файл.
 - c - create, създаване на архив
 - t - показва съдържанието на архива
- x-extract - разархивира
- gcc - компилатор

gcc source.c //комп. + изл. файл. /out gcc source.c -o opt //комп. изл. файл. /opt

Тема 12

1) Процес и неговата локална памет - методи за използване и защита.

Иерархия на паметите - кеш, RAM, swap.

Виртуална памет на процеса - функционално разделяне (програма, данни, стек, heap, споделени библиотеки).

• Кеш - скопии, кои бързи

- подходящи за пресмятане; поддържа данните на паметта
- спомагателна памет за ускоряване обмена на данни между различните нива в иерархията на паметта. Ускоряването се постига чрез поддържане на копия от избрани части от данните върху постепен с бързо действие, близко до това на горното ниво на паметта. Може да постигне различни степени на ускорение в зависимост от вида на обменяните данни, от алгоритъм за избор на данните за копиране и от обвместимостта помежду членовете на паметта, които са сравнително малки.

• RAM - по-голямата част от процеса е тук

• Паметта, която процесорът директно използва.

• В нея се разполагат стека, кода на програмата, данните.

• Swap - памет върху диска, която е част от диска

• когато части от процеса са блокирани по-бавни хардуерни ресурси
• нарича се swapping - временно съхранение на редко използванияте програми.

• Виртуална памет - системна памет, симулирана от ОС и разположена на твърдия диск. Тя позволява да се прилага едно и също, непрекъснато адресиране на физически различни памети.

⇒ Програма

→ Данни - глобални променливи, константи, статични променливи от програмата.

→ Стек - използва се от програмата за променливи и запазване. Расте и се създава размера в зависимост от това какви практики са извикани и какви са техните изисквания за размера (в размер?).

→ Heap - използва се за някои видове работни хранилища.

→ Споделени библиотеки - използувано независимо и така те могат да бъдат споделени от всички програми, които искат да ги използват

Тема 13

1) Таблица за съответствието виртуална/реална памет.

Хардуерът, който отговаря за трансляцията между логически и физически номер на страницата и свободе за управлението на адресацията, се нарича ММИ (Memory Management Unit). Инструментът, който използва за оптимизиране на работата на ММИ, се нарича TLB (Translation Lookaside Buffer). В зависимост от схемата за трансляция, TLB може да се реализира по различни начини.

Ефективна обработка на адресацията - ММИ, TLB

- ММИ-хардуер, през който всички редференции на паметта преминават като главно избриват трансляцията от виртуален адрес във физически. Модерните ММИ по принцип разделят виртуалното адресно пространство в страници, всяка от която има размер, степен на 2. Най-долните битове на адреса остават непроменени. Горните битове на адреса са числата на виртуалната страница.
- TLB-кеш, който участва в управление на паметта използва, за да подобри скоростта при трансляция на виртуални адреси. TLB има фиксиран брой слота, съдържащи записи от таблицата със страници и от таблицата със сегменти. Записите от таблицата със страници се използват за преобразуване на виртуалните адреси във физически адреси, докато записите от таблицата със сегменти преобразува виртуалните адреси в сегментни адреси.

2) файлови дескриптори, номера на стандартните fd, пренасочване

Програмата при стартиране има 3 отворени файла - stdin(0), stdout(1), stderr(2). В самия език такъв отворен файл се

нарича файлов дескриптор, представен чрез число от 0 до 2. Коявър, данните като изход от 1 се пренасочват като вход на 2.

< - пренасочване на стандартния вход

> - пренасочване на стандартния изход (ако файлът не съществува, иначе пренасочва)

>> - пренасочване на стандартния изход като се пренасочва, а добавя.

2> пренасочване на стандартния изход за грешки

2>> пренасочване на стандартни изход за променят первоначалните

фильтри-команди, които обработват данните без да променят первоначалните

• cat cat [f1-fn] - показва съдържанието на файловете

• grep grep str file - търси иззърв. израз във файл; върши редове от файла, в които съдържа

• cut cut [options][file] - търси редове от файл -c (символ); -f (номер на колона); -d (разделител)

• sort sort f1 - сортира f1 по редове (-o output-file - резултатът се запиши във файл)

• wc - извежда ико за брой редове(-l), байтове(-c), думи(-w) • tr - заменя зетачи на други конкретни символи.

Тема 14

1) Избройте видове събития, причиняващи повреда на данните във файловите системи.

Описете кратко стандарта RAID 5.

Какво е журнална файлова система?

- дефект в хардуера
- софтуерна грешка
- грешка на потребителя, интерфейса
- вируси
- спирате на тока

→ RAID 5 е механизъм, който предотвръща загуба на информация. При него се използват няколко диска, например 5. Диските се разделят на сектори като за всеки от секторите пишат на един от диска в същия сектор сумата на останалите 4 по модул 2 (контролна сума). Така, като и диск да се отключи, може да се възстанови файловата сума, която проследява последиците на промените в структурата на данни известна като рутини (функции) поддържащи промените в структурата на данни известни като рутини (функции).

→ Журнална файлова система - всички современи файлови системи са всички операции съхранявани във файла (или част от файла) са обработвани за да съхранят правилния ред в журнала.

2) Съвръзване и допускане до UNIX системата - login.

Конзола - стандартен вход, изход, стандартна грешка.

Команден интерпретатор - shell. Изпълнение на команди, параметри на команди.

Тема 15

1) Опишете разликата между синхронни и асинхронни ви операции.
Дайте примери за програми, при които се налага използването на асинхронен вход-изход.

• Синхронна ви операция:

Процесът проверява дали ресурсът е свободен:
↳ ако е => процесът получава нужните данни;
↳ ако не е => процесът се приснива (блокира), изпълждането става, когато ресурсът е свободен;

• Асинхронна ви операция:

Процесът проверява дали ресурсът е свободен:
↳ ако е => процесът получава нужните данни;
↳ ако не е => процесът не се приснива, а продължава да върши никаква друга работа и ядрото временно управлява процеса със специален код за гримка, който спути за определяне степента на завършеност на операцията.

Примери:

(1) При използване на WEB-Browser той трябва да реагира на входни данни от клавиатура и мишка, както и на данни, постъпващи от интернет, т.е. на 3 входни канала. Браузерът чрез асинхронни опции за четене проверява по кой от каналите постъпва информация и реагира адекватно.
(2) Серъверът в интернет може да обслужва много на брой клиентски програми като поддържа отворени TCP връзки. За да обслужва паралелно клиентите, серъверът трябва да ползва асинхронни операции, за да следи по коя връзки протича информация и коя са пасивни.

(2) Опишете с по 1-2 изречения работата на следните извиквания:

- `socket(domain, type, protocol)` - отварява сокет (връзка между процеси)
- `bind(sfd, &my-addr, addrlen)` - присваива име на сокета (което не са в роднинска връзка)
- `listen(sfd, backlog)` - започва приемане на заявки за връзки.
- `connect(fd, &server-addr, addrlen)` - подава заявка за изпратване на връзка между именувания сокет sfd и fd.
- `accept(sfd, &peer-addr, addrlen)` - приема заявка за изпратване на връзка => изпредадена е връзка между ф.у sfd на серъвера и fd на клиента.

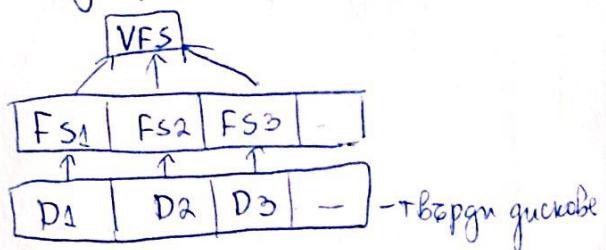
Тема 16

1) Опишете понятието "пространство на имената".
Как изглежда това "пространство в OS Linux"?

Пространството на имената са всички дълготрайни обекти в системата. Още се нарича виртуална файлова система.

Тази виртуална файлова система е кореново дърво.

Чрез това дърво можем да добавяме устройство с mount и да премахваме чрез umount.



2) Една от класическите задачи за синхронизация се нарича "Задача за читателите и писателите" (readers-writers problem).
Опишете условието на задачата и решение, използващо семафори.

Проблемът в задачата с читателите и писателите се отнася до всички ситуации, в които общи ресурси се четат и се модифицира от конкуриращи се процеси. Докато в структурата от данни се пише или се модифицира е необходимо да се задържат на другите читачи/процеси да я четат, за да се предотврати прекъсването на писането, както и прочитането на нещо невалидно.

Синхронизационни ограничения:

- (1) Наколко читатели могат да четат в ресурса едновременно.
(2) Ако има писател в ресурса тој е единствен.

Reader

```
Barrier.wait()  
Barrier.signal()  
mutex.wait()  
cnt = cnt + 1  
if cnt == 1  
    roomEmpty.wait()  
mutex.signal()  
  
READ  
mutex.wait()  
cnt = cnt - 1  
if cnt == 0  
    roomEmpty.signal()  
mutex.signal()
```

Writer

```
barrier.wait()  
roomEmpty.wait()  
  
WRITE  
barrier.signal()  
roomEmpty.signal()
```

semaphore

roomEmpty.init(4)

int cnt=0

semaphore mutex.init(1)

защита на бранка
(избягване race condition)

semaphore barrier.init(1)

избягване издаването на
писатели, запади много
читатели.

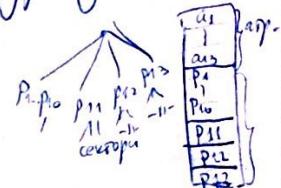
Тема 17

1) Описете какви атрибути имат файловете в съвременната файлова система, реализирана върху блочно устройство (block device).

Основните атрибути на файла са име, права за достъп, дата на създаване, дата на последна промяна.

Има още размер, собственик, парола за достъп и флагове.

В съвременната OS в директориите файловете се изразяват чрез двойка < , >, сочи към сектор



(a) отлагане на записа, алгоритъм на асансьора

• всички съвременни файлови системи са журнали (някакъв специален файл (или част от диск) се добавява за журнал и отложените операции се записват в правилния ред в журнала). Отлагането на записа се прилага с цел бързодействие.

• Алгоритъм на асансьора - предполага, че близки сектори на диск не се променят бързо. При него чинаме указател, който сочи къде се намира главата на твърдия диск (в момента, когато се и посоката, в която се движи тя. Заявките се обработват по посока на главата подобно на движението на асансьор - спира надолу докато изпълни всички заявки, след това нагоре, докато изпълни всички)

Недостатък може да доведе до голямо задавяне на заявките.

(b) поддържане на буферизацията на файловата система

Кешът на файловата система съдържа данни, които чакат да са прочетени от диск, което позволява на следващите заявки да получат данни от кеша, вместо да се налага да ги чете отново от диск.

2) Как се изгражда ком. канал (connection).

Процес - сервер:

- 1) $sf = \text{socket}(\text{domain}, \text{type}, \text{protocol})$
- 2) $\text{bind}(sf, \&\text{my-addr}, \text{addrlen})$
- 3) $\text{listen}(sf, \text{backlog})$
- 4) $cf = \text{accept}(sf, \&\text{peer-addr}, \text{addrlen})$

създава сокета sf и 2) го именува.

процеса за изграждане на връзки.

създава сокета fd без да е нужно да го именува.

и) клиентът

заявка за изграждане на връзка към имен. сокет sf.

Процес - клиент:

- 4) $fd = \text{socket}(\text{domain}, \text{type}, \text{protocol})$
- 5) $\text{connect}(fd, \&\text{server-addr}, \text{addrlen})$

6) Серверът приема заявката.
Изградена е връзка
нр. ф.г. сfd на с.
и fd на клиент
изр. ги за
общен за
информационния

Тема 18

1) Опишете кратко основните комуникационни канали в OS Linux. Кои канали използват пространството на имената и кои не го правят?

Основните комуникационни канали в OS Linux са тръба (pipe), иметубка (fifo), връзка (socket) и конекция (установена чрез механизъм socket).

Всички освен обикновената тръба използват пространството на имената. Операците за ползване на канала са общни - read(), write(), close(). Специфични са изискванията за изграждане на различните видове канали.

2) Опишете какви изисквания удовлетворява съвременната файлова система, реализирана чрез блочни устройства (block device)?

Блочните устройства комуникират като изпращат член блок от данни (твърди дискове, USB камери). Всички устройства предлагат буферизиран достъп до хардуерни устройства и предлагат някаква абстракция от тяхната специфика. Предимство - програмистът чете и пише блок от всички размер и правилното използване.

Недостатък - не знаем какво време ще отнеме прехвърлянето от адреса в истинското устройство или в какъв ред.

Опишете предназначението на журнала във файловата система.

Журналната файлова система проследява недописаните промени като поддържа промените в структура от данни (журнал/цикличен буфер). Когато редим да запишем файл, то файловата система може (1) да отбележи в журнала, че иска да запише файл

(2) да запише файла

(3) да отбележи в журнала фактът че файлът е успешно записан

(4) да регистрира файла

(5) да изтрие журнала

Ако захранването спре, при следващото зареждане файловата система първо ще прочете журнала и ще пречести какво да предприеме, като по този начин ще ни остави последователна файлова система