# Устен изпит ОС – Теми

*Подбраното е от доста различни източници, може да има пропуски/несъответствия. Със зелено е отбелязано най-важното за темата – Наско*



## 1.

**Тема 1**

Опишете как ОС разделят ресурсите на изчислителната система, дайте примери за основните типове разделяне:
Разделяне на пространството (памети).
Разделяне на времето (процесори, други у-ва).

Опишете с по едно-две изречения работата на следните системни извиквания в стандарта POSIX:
pipe() dup2() fork() exec() wait() waitpid()

Разделяне на пространството (памети).

Когато един процес работи, неговите данни и стек трябва да бъдат негова монополна собственост, т.е., никой друг да няма достъп до тях. Тази защита на данните се осигурява хардуерно.

Има два основни режима на работа:

Kernel mode (ядрото): програмата има достъп до цялата памет (всички ресурси).

Protected mode (обикновен процес): програмата има достъп само до паметта, която й е предоставена, както и до някои инструкции.

Единият вариант това да се случи е чрез сегментация. За паметта, която е предоставена на всеки процес, имаме два регистъра: единият указва началото f, а другият дължината s, и проверяваме за всеки байт, използван от текущия процес, дали е в интервала [f, f + s).

Друг вариант е paging. Имаме таблица, която за всяка страница указва какви права има текущият процес.

Съвременната концепция е всеки потребителски процес да си има такава таблица. Така той разполага с памет, която имитира рам паметта, а в адресната таблица се пазят реалните адреси на страниците, с които процесът работи. Те могат да са разположени на рам паметта или на твърдия диск. Когато се изчерпа мястото на рам паметта, страници(сегменти), които не се използват, се местят на твърдия диск.

При разпределянето на процесорното време голяма роля играе типът на процеса:

- I/O: (Foreground) Прави входно-изходни операции, обикновено чака да прочете данни от някакъв буфер и да ги запише някъде. Изпълнява се с приоритет за бързина на входно-изходните операции. Често бива приспиван, прекарва много време между Running и Active и е важно когато бъде събуден много бързо да получи процесорно време.
- CPU: (Background) Прави изчислителни операции, които могат да отнемат много време (т.е., няма често да се приспива), но няма условие кога да бъдат завършени.
- Real-Time: Прави важни операции, включващи например комуникация с външни процеси/устройства, което изисква бърз отговор в даден срок. Използва deadlines, които налагат този процес да се приоритизира така, че да изпълни задачата си в зададеното време.

В контекста за разделяне на процесорно време най-простият алгоритъм е round-robin. Процесите се подреждат в обикновена опашка и никой няма по-голям приоритет от другия. Така се предотвратява гладуване, но имаме проблеми с I/O процесите и Real-Time процесите.

В съвременните ОС се използват различни методи за присвояване на приоритети: може например да има плаващи приоритети на процесите и всеки път, когато процесорът чака процесорно време и не му се дава се увеличава неговият плаващ приоритет и се взима максимумът измежду константният и плаващият приоритет. Друг вариант първоначално да се дава един и същи приоритет и да се вади от приоритета на процеса процесорното време, което той е използвал в текущия такт.

- Pipe() – създава се комуникационен канал FIFO. Приема int fd[2] и връща два файлови дескриптора – за четене и за писане.
- Dup2() – приема първи параметър файлов дескриптор oldfd и втори – newfd. Затваря втория (ако е сочел към нещо) и го дублира на първия.
- Fork() – създава копие на текущия процес, което е на същата стъпка от прочитането на инструкциите, на която е и родителският процес. Детинският процес наследява всички отворени файлови дескриптори, променливи и т.н.. Връща 0 ако се чете от детинския процес и pid на детинския процес ако се чете от родителския.

- Exec() – семейството от системни извиквания приема shell команда и произволен брой параметри под формата на символни низове, терминирани с NULL. Изпълнява съответната команда като заменя текущия код на програмата с нея, т.е., променя програмата, която текущият процес изпълнява.
- Wait() – родителски процес чака даден детински процес да приключи своето изпълнение. Може да приеме параметър, който да получи данни за изходния статус на детинския процес.
- Waitpid() – родителски процес чака процес с pid, подаден параметър, да приключи своето изпълнение. Wait & Waitpid връщат pid на терминиралия детински процес.

2.

**Тема 2**

Опишете разликата между времеделене и многозадачност.
Какви ресурси разделя еднозадачна, еднопотребителска ОС?

Опишете с по едно-две изречения работата на следните системни извиквания в стандарта POSIX:
open() close() read() write() lseek()

Многозадачност наричаме способността няколко процеса да работят паралелно. Когато някое задание чака за изпълнение на входно-изходна операция, процесорът може да премине към друго задание.

Времеделение наричаме „нарязването“ на процесорното време на малки интервали като на всеки процес се дава такъв интервал. Така ако имаме няколко потребители и процесорът е един, се създава илюзията, че за всеки потребител има отделен процесор.

Основната разлика е, че времеделението е споделяне на изчислителен ресурс между много потребители чрез едновременна многозадачност, а многозадачността е едновременното изпълнение на множество задачи или процеси в даден момент от време. Докато времеделението позволява на много потребители да използват компютърната система едновременно, многозадачността позволява на много процеси да използват процесора наведнъж.

Еднозадачната еднопотребителска ОС е ОС, която позволява на един потребител да върши една задача в даден период от време. Заема по-малко място на паметта и използва целия си ресурс за този един потребител и тази една задача.

- Open() – създава файлов дескриптор, който сочи към реферирания в първия аргумент адрес в пространството на имената (файловата система). Втората секция на аргументи представлява побитово ‚ИЛИ‘ на произволен брой флагове, които определят дали файлът ще се чете и/или пише, дали да бъде изтрит, ако вече съществува, дали да бъде създаден, ако не съществува, дали да се премести курсорът в края на файла и т.н.. Третият (опционален) аргумент е осмично число, което указва правата върху файла (когато той се създава).
- Close() – затваря подадения като аргумент файлов дескриптор и освобождава този ресурс от паметта.
- Read() – приема аргумент файлов дескриптор, от който ще се чете, указател към структура (памет), в която ще се запишат прочетените данни, и размер в байтове на данните, които искаме да прочетем. Връща броя прочетени байтове.
- Write() – приема аргумент файлов дескриптор, в който ще се пише, указател към структура (памет), която ще се запише в реферирания от дескриптора файл, и размер на данните, които искаме да запишем. Връща броя записани байтове.
- Lseek() – мести указателя за четене/писане на указано място. Първият аргумент е файлов дескриптор, в който ще местим указателя, вторият е отстояние (в байтове), а третият е SEEK_CUR, SEEK_SET или SEEK_END. Връща мястото във файла (поредния байт от началото), където сме преместили курсора.

3.

**Тема 3**

Дайте кратко определение за: многозадачна ОС, многопотребителска ОС, времеделене.

Опишете разликата между многопотребителска и многозадачна работа. Какви качества на ОС характеризират тези две понятия?

Опишете с по едно-две изречения работата на следните системни извиквания в стандарта POSIX:

open() close() lseek() pipe() dup2()

Многозадачна ОС – може да изпълнява множество задачи (да действат множество процеси) едновременно.

Многопотребителска ОС – една система може да се използва от множество потребители едновременно.

Времеделене – ОС позволява на множество потребители да работят върху нея едновременно (създавайки илюзията, че за всеки потребител има един процесор), като процесорното време се „нарязва“ на малки парчета, които се разпределят между процесите.

Основната разлика е, че многозадачната ОС не е гарантирано многопотребителска ОС(но може да поддържа работата на множество процеси едновременно), докато многопотребителската ОС може да се използва от произволен брой потребители едновременно.

Качества – многопотребителска работа, разпределена обработка и управление на ресурсите, мултипрограмна работа.

4.

**Тема 4**

Опишете ситуацията съревнование за ресурси (race condition), дайте пример.

Опишете накратко инструментите за избягване на race condition:
(а) дефинирайте критична секция, атомарна обработка на ресурса.
(б) инструменти от ниско ниво, специфични хардуерни средства.
(в) инструменти от високо ниво, които блокират и събуждат процес.

Каква е спецификата на файловете в следните директории в Linux:
/etc /dev /var /boot /usr/bin /home /usr/lib /var/log

Race condition наричаме явлението, при което два или повече процеса/нишки достъпват споделени данни, което може да доведе до разрушаване на данните. Пример (read-modify-write): имаме променлива i, която искаме да инкрементираме с единица, но няколко процеса имат едновременно достъп до този блок код:

i++

Тук на практика се извършват три операции и е възможно единият процес да присвои на i стойност, променена в същия момент от другия процес.

Критична секция наричаме място, където се достъпват споделени данни от повече от един процес едновременно.

Атомарната обработка на ресурс представлява операция, която се изпълнява без прекъсване (като единна, неделима инструкция) от операционната система (в един единствен такт).

Инструментите от ниско ниво, специфични хардуерни средства:

- Spinlock – инструмент, който посредством ключалка (бит, който е 0 за свободна структура и 1 за заета) кара процесите да чакат в цикъл докато не се освободи ключалката
- Атомарни операции test-and-set, atomic swap, atomic increment
- Enable/Disable interrupt – инструмент за контрол над блокирането на прекъсванията на процесите

Инструментите от високо ниво – Семафори. Семафорите са механизъм, който следи броя процеси, които имат достъп до данните. Когато процес получи

достъп до данните, този брояч се декрементира и в случай, че броячът е 0, процес, който заяви достъп, бива приспан.

Специфики на файловете в следните директории:

- /etc – тук се пазят данни за акаунтите на потребителите – информацията, свързана с тях, пароли и т.н.
- /dev – описват се устройствата, които са част от изчислителната система, информация за драйверите на устройствата
- /var – съдържа споделени данни (напр. бази данни)
- /boot – съдържа информация, касаеща стартирането на работата на операционната система при зареждане
- /usr/bin – всички програми, които са изпълними процеси (двоични файлове)
- /home – тук са потребителските директории на всеки един от потребителите – персоналната част от ОС
- /usr/lib – включва обектни файлове и библиотеки (изходен код)
- /var/log – Файлове за регистрация, автентикация, инсталация/деинсталация, системни файлове, апликационни файлове

5.

**Тема 5**

Хардуерни инструменти за защита (lock) на ресурс:
(a) enable/disable interrupt
(b) test and set
(c) atomic swap
Опишете инструмента spinlock, неговите предимства и недостатъци.

Каква е спецификата на файловете в следните директории в Linux:
/etc /dev /var /proc /bin /home /usr/doc

- Enable/disable interrupt е механизъм, който позволява да се блокират прекъсванията на процесите. Прекъсване може да се получи например при входно-изходни операции или зададен таймер за чакане. С този механизъм се позволява на процеса да игнорира такива съобщения за

прекъсване и да продължи работа. Има ключово значение при управление на критични секции, защото гарантира, че процесът в критичната секция няма да бъде спрян от външни събития, което би могло да доведе до race condition.

- Test-and-set е атомарна операция, която проверява стойността на дадена променлива, модифицира я и връща оригиналната стойност.
- Atomic swap е атомарна операция, което проверява дали дадени данни в паметта съответстват на някаква стойност, и само в случай, че това е така, модифицира тези данни.

Spinlock е инструмент от ниско ниво за управление на критични секции. Състои се от допълнителен бит (ключалка), който е 0 когато критичната секция е свободна, и 1 когато е заета. Когато има процес, който използва критичната секция, и се появи друг процес, който иска достъп до нея, той чака в цикъл докато не се освободи ключалката. Spinlock е подходящ за кратки критични секции, когато те са дълги има голяма вероятност да се предизвика гладуване на чакащите процеси. Предимствата на spinlock се изразяват в ефикасността му в предотвратяването на race condition-и. Недостатък на този метод е, че процесите, които циклят (busy wait), не могат да вършат друга полезна работа докато чакат, съответно се губи производителност. Други недостатъци – може да се получи безкраен цикъл при извикване на процес, който също има нужда от споделения ресурс, може да се получи зацикляне на процесора при прекъсване вътре в критичната секция (затова трябва да се забраняват прекъсванията преди критичната секция).

Специфика на файловете в следните директории:

- /proc – съдържа информация за всички активни процеси в ОС
- /bin – съдържа изпълними файлове, важни програми, необходими за стартиране на ОС, общодостъпни програми, реализиращи команди
- /usr/doc – централна директория за документация – съдържа цялата необходима документация в ОС

6.

**Тема 6**

Опишете понятията приспиване и събуждане на процес (block/wakeup).
Семафор – дефиниция и реализация.
Опишете разликата между слаб и силен семафор.

Опишете накратко различните видове специални файлове в Linux:
външни устройства, именувани в /dev, псевдофайлове в /proc
линкове – твърди и символни, команда ln
сокети

Приспиване и събуждане на процес (block/wakeup). Когато даден процес чака определено събитие да настъпи, независимо дали е входно-изходна операция, или времеви момент, или нещо друго, той бива приспан. Обикновено процесът се приспива сам чрез системното извикване block(), а по-късно, когато настъпи конкретното събитие, той бива събуден от друг процес/сигнал (wakeup).

Семафорът е механизъм за (синхронизация) предотвратяване на race condition-и, който може да приспива чакащите процеси, когато една критична секция е заета, и да събужда чакащ процес когато тя се освободи. Състои се от брояч на процесите с достъп до ресурса и списък (контейнер), който съдържа спящите процеси. Когато броячът е $< 0$, абсолютната му стойност показва броя приспани процеси в контейнера.

Има следните методи:

- Init – инициализира семафора с дадена начална стойност на брояча (колко процеса първоначално могат да използват критичната секция)
- Wait – поставя се преди критичната секция – когато броячът е $> 0$, чакащият процес се допуска до критичната секция, а когато е 0 или по-малък, процесът се приспива (добавя се към контейнера с приспани процеси)
- Signal – когато един процес вече е напуснал критичната секция, той вече може да уведоми чакащите чрез този метод – той увеличава брояча с единица и може да събуди един от спящите процеси и да го допусне до секцията.

Силен семафор е този, при който контейнерът с чакащи процеси се реализира като обикновена опашка (FIFO), докато слабият семафор поставя определени (различни) приоритети на процесите в своя контейнер.

Видове специални файлове в Linux:

- Външни устройства, именувани в /dev – псевдофайлове, които съответстват на периферни устройства – задават какъв хардуер може да се обслужва от ОС и какъв точно се обслужва
- Псевдофайлове в /proc – това е виртуалната файлова система – съдържа статистическа информация за активните процеси, връзките между тях и т.н.
- Линкове – това са файлове, които по някакъв начин реферират към друг файл. Твърдите символни линкове имат същия inode като оригиналния файл, съответно споделят неговата памет, а символните линкове представляват указател към мястото, където е записан оригиналният файл
- Командата ln създава линкове към даден файл, който се подава като първи аргумент. Като втори аргумент се подава името на съответния линк, като по подразбиране се създава hardlink, но с опцията -s може да се създаде symbolic link
- Сокети – файлове, които пазят информация като IP адрес и номер на порт за свързване – служат за изграждане на комуникационен канал по мрежата.

7.

**Тема 7**

Взаимно изключване – допускане само на един процес до общ ресурс. Опишете решение със семафори.

Качества и свойства на конкретните файловите системи, реализирани върху block devices.

Ефективна реализация, отлагане на записа, алгоритъм на асансьора.

Взаимно изключване (Mutual exclusion или mutex) – Метод за синхронизация, при който само един процес/нишка в даден момент има достъп до споделените данни.

Примерно решение със семафори:

Semaphores: mutex

mutex.init(1)

mutex.wait()

{ critical section }

mutex.signal()

Качества и свойства на конкретните файлови системи, реализирани върху block devides:

Всяка директория във файловата система съдържа обекти (файлове) от типа <име, inode>, където inode (index node) е блокът, който съдържа информацията за файла, както и указатели към блоковете със съдържанието му, а името на файла е абсолютен път до него. Когато четем някакъв файл, ние отваряме негова твърда връзка и съответно този файл може да се ползва на много места едновременно (и да се модифицира) без това да повлияе на инстанцията, която четем.

Ефективна реализация – твърдият диск е диск, разделен на много пътечки.

- Над повърхнината има глава, която засича дали битът е 1 или 0
- Главата не се плъзга по повърхността, а лети много ниско над нея
- Преминаването на главата от една пътечка до друга е бавна механична операция
- Всяка пътечка се разделя на сектори с дължина 512 или 1024 байта (блокове)
- Софтуерът знае колко е голям един блок и колко време е нужно на главата да се придвижи от един блок към друг
- Когато се създава един файл, той не се разполага върху последователни блокове, а се „разхвърля“ на различни свободни места (за да може да увеличава размера си). Този процес се нарича фрагментация

Отлагане на записа – заявките за промени върху файловете във файловата система се записват в т.нар. журнал (log файл). Когато този журнал се запълни, започват да се обработват действително заявките за входно-изходни операции. Ако спре захранването, информацията за необработените заявки все още ще се пази в журнала.

Алгоритъм на асансьора (SCAN) – когато събира заявките, четящата глава трябва да измине възможно най-краткия път. Заявките се разместват така, че да бъде изминато най-краткото разстояние. Правят се две приоритетни опашки – една с файлове в посоката на движение на главата и друга – в обратната посока. Ако се изпразни опашката по посоката на движение на главата, се сменя посоката.

8.

**Тема 8**

Комуникационна тръба (pipe), която съхранява един пакет информация – реализация чрез редуване на изпращача/получателя.

pipe с буфер – тръба, съхраняваща n пакета информация. Използване на семафорите като броячи на свободни ресурси.

Права и роли в UNIX, команда chmod
Права – u/g/o user/group/others
Роли – r/w/x read/write/execute

Тръбата, която съхранява един пакет информация, се използва като се редуват операциите четене и писане върху нея. Изпращачите на данни пишат в тръбата, а през това време получателите чакат да вземат данните от нея. Тази операция може да се повтаря множество пъти в двете посоки. Реализация чрез семафори:

Semaphores: free_bytes, ready_bytes

free_bytes.init(1)

ready_bytes.init(0)

| Producer: | Consumer: |
| --- | --- |
| free_bytes.wait() | ready_bytes.wait() |
| { write to buffer } | { read from buffer } |
| ready_bytes.signal() | free_bytes.signal() |

Тръбата, която може да съхранява n пакета информация, се реализира чрез опашка, в края на която изпращачите поставят данни, а получателите ги взимат от другата страна (началото), за да ги обработят. Реализация чрез семафори:

Semaphores: mutex_read, mutex_write, free_bytes, ready_bytes

Data structures: Queue

Free_bytes.init(n)

Ready_bytes.init(0)

Mutex_read.init(1)

Mutex_write.init(1)

| Producer: | Consumer: |
| --- | --- |
| free_bytes.wait() | ready_bytes.wait() |
| mutex_write.wait() | mutex_read.wait() |
| { add to Queue } | { retrieve from Queue } |
| mutex_write.signal() | mutex_read.signal() |
| ready_bytes.signal() | free_bytes.signal() |

Права и роли

Права – всеки обект от файловата система (файл) се асоциира с определен набор от права за достъп, които определят интеракциите с него. Те са три

типа – права за създателя на файла (user), права за групата, към която принадлежи потребителят (group), и права за всички останали (others).

Роли – за всеки от трите типа права има три основни вида роли – read, write и execute. Те бележат съответно разрешенията за четене, писане и изпълняване на дадения файл.

Съвкупността от права и роли може да се дефинира чрез осмично число, като за всяка тройка от роли позициите в осмичното число са следните: $2^2$ (4) маркира възможност за четене, $2^1$ (5) маркира възможност за писане и $2^0$ (1) възможност за изпълнение.

Обикновените файлове се създават по подразбиране с права 666, а директориите – със 777. И на двата типа обекти се налага umask – битова маска със стойност 0022 по подразбиране.

chmod – команда, която служи за променяне на правата на даден обект (файл). Приема като първи аргумент правата, с които ще актуализираме старите (в human-readable формат, или в осмичен вид), и като втори аргумент път до самия файл.

9.

**Тема 9**

Взаимно блокиране (deadlock)
Гладуване (livelock, resource starvation)
Пример: задача за философите и макароните

Единна йерархична файлова система в UNIX.
Файлове и директории, команди – cd, mkdir, rmdir, cp, mv, rm

Deadlock се нарича явлението, при което даден процес чака за достъп до ресурс, който никога няма да получи. За наличие на Deadlock са необходими 4 условия:

- Един процес има ексклузивни права върху ресурса, до който в момента има достъп

- Процес може да заяви желание за достъп и до други ресурси докато все още държи тези, до които в момента има достъп
- Не може насила да се отнеме ресурса от даден процес
- Налична е кръгова опашка от два или повече процеса, всеки от които търси ресурс, който се държи от предходния

Гладуване – явление, при което процес е лишен от даден ресурс и не може да прогресира в работата си

Задачата за философите и макароните: Проблемът описва кръгла маса, около която са седнали петима философи, всеки от които има своя чиния, но вилиците са също 5 – между всяка чиния на масата. Всеки има нужда от две вилици, за да се нахрани. Ако всеки от тях вземе първо дясната вилица, то за всеки философ ще има точно по една вилица (когато посегне за лявата, тя вече ще бъде заета от съседа)  и ще се стигне до deadlock.

В UNIX файловата система е йерархична  (дървовидна). Съставена е от множество директории. Коренът на дървото е ‚/‘. Файлът е основната единица за съхранение на данни. Директорията е структура, чрез която дадена съвкупност от файлове могат да се обособяват по даден критерий.

Типове файлове – директория, обикновен файл, символен линк, наименувана тръба, сокет.

Команди:

- cd – преместваме се в посочения относителен/абсолютен път от файловото дърво
- mkdir – създаваме празна директория по подаден път
- rmdir – изтриваме празна директория по подаден път
- cp – копираме файл от посочения път в избран нов път (където може да има ново име)
- mv – местим файл от посочения път в избран нов път (където може да има ново име)
- rm – изтриваме файл, който съответства на посочения път (с опцията -r изтрива директория, която може да не е празна)

10.

**Тема 10**

Процеси в многозадачната система.
Превключване, управлявано от синхронизация.
Превключване в система с времеделене – timer interrupt.

Опишете функционалността на следните команди в Linux:
ls, who, find, ps, top

Процесите в една многозадачна система биват:

- I/O (Foreground) – процеси, които завършват работата си бързо, през по-голямата част от времето чакат (често са в режим Ready)
- CPU (Background) – процеси, които изпълняват големи изчислителни задачи, които отнемат повече процесорно време. Няма значение кога точно ще бъдат завършени тези операции
- Real-Time – процеси, които вършат важни задачи с някакъв установен deadline – трябва задължително да приключат успешно работата си в даден момент.

Превключването, управлявано от синхронизация, се изразява в приспиването на процеси (чрез wait на семафор), когато стигнат до критична секция, и съответно събуждането им, когато друг процес вече е изпълнил дадена операция и секцията вече е свободна.

Превключването с timer interrupt се случва когато даден процес прекара определен период от време в процесора и се налага да се освободи CPU ресурс. Това е така заради времеделенето – на всеки процес се дава определен допустим времеви интервал, в който може да ползва процесорно време. След като изтече таймерът, процесът се приспива и процесорното време се дава на друг(следващият минава от Ready в Running).

Команди:

- ls – извежда списък от обекти в дадена директория от файловата система
- who – извежда списък от активни потребителски сесии в момента на извикване

- find – търси в подадена като първи аргумент директория дадени обекти (файлове), които може да филтрира чрез богат набор от критерии като тип, име, размер и т.н.
- ps – извежда списък с активните процеси в момента на извикване
- top – извежда интерактивен списък с работещите процеси в реално време

11.

**Тема 11**

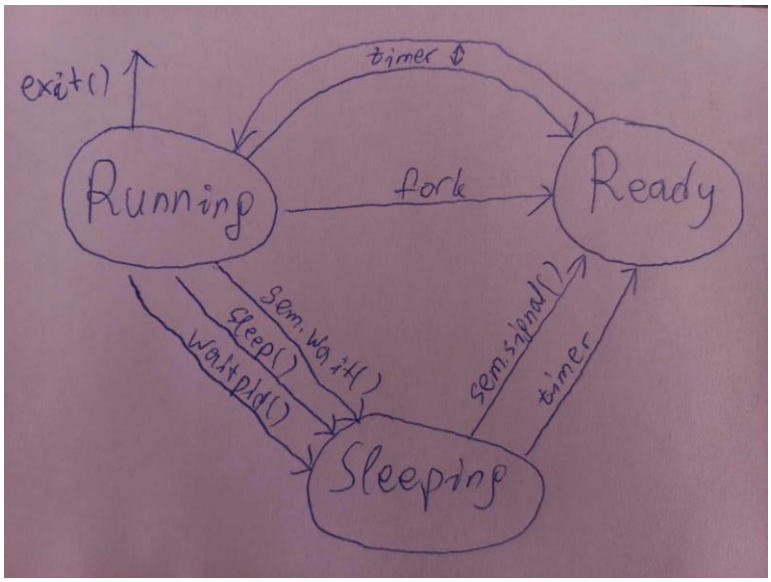
Възможни състояния на процес. Механизми и структури за приспиване/събуждане.
Диаграма на състоянията и преходите между тях.

Опишете функционалността на следните команди в Linux:
vi, tar, gcc

/* Тази ми се падна на мен и на първата част имах 15/20, не знам защо */

Възможни състояния на процес:

- Running – в момента използва процесорно време
- Ready – готов е за работа, но все още не му е дадено процесорно време
- Sleeping – блокиран – очаква дадено събитие:
  - I/O – очаква входно-изходна операция
  - Timed wait – очаква времеви момент
  - Signal – очаква сигнал от друг процес
  - Страницата, върху която иска да работи, се намира някъде на твърдия диск
- Stopped – спрян
- Zombie – в процес на спиране (спирането на процес е бавна операция)




Команди:

- vi – стартира изпълнение на текстовия редактор vi, може да се подаде файл като аргумент (за обработка)
- tar – в режим на създаване (-c) архивира подадения като втори аргумент файл (директория) и го записва в подадения като първи аргумент път. В режим на екстракция (-x) разархивира подадения като аргумент файл. Поддържа набор от компресиращи алгоритми, с които файлът, освен да се архивира/разархивира, може и да се компресира/декомпресира .
- gcc – компилира даден изходен файл (на езика C), при успешна компилация се създава изпълним файл, чието име можем да модифицираме с опцията -o.

12.

**Тема 12**

Процес и неговата локална памет – методи за изолация и защита.
Йерархия на паметите – кеш, RAM, swap.
Виртуална памет на процеса – функционално разделяне (програма, данни, стек, heap, споделени библиотеки).

Опишете функционалността на следните команди в shell:
echo, read, test, if, for, while

Процес и неговата локална памет – методи за изолация и защита – за да се изолират процесите един от друг, се използват механизми като виртуална памет и сегментация, чрез които се обособява памет за всеки един процес.

Кеш – спомагателна памет за ускоряване на обмена на информация. Скъпа, но бърза. Ускоряването се постига чрез поддържане на копия от избрани части на данните върху носител с бързо действие. За по-голяма ефикасност кешовете са с малък размер.

RAM памет – паметта, която директно се ползва от процесора. По-голямата част от процеса е тук. Тук се разполагат стекът, кодът на програмата, данните и т.н.

Swap – механизъм, чрез който рядко използвани ресурси се местят временно на специално място на твърдия диск, за да се освободи място на главната памет.

Виртуална памет – системна памет, която се симулира от ОС, позволява да се прилага непрекъснато адресиране на физически различни памети (участъци от твърдия диск).

Виртуалната памет бива:

- статична (read-only) – съдържа изходния код на програмата, която се изпълнява, както и споделени библиотеки, които могат да се ползват от много процеси едновременно
- динамична (read-write)
    - стек – съдържа важни структури за управление на програмата (функции, локални променливи и т.н.). Може да расте и намалява в зависимост от размера на използваните структури.
    - heap – използва се за някои видове работни хранилища (структури от данни)
    - блок за статична памет – място за константи, глобални (статични) променливи – данни, за които при всяко стартиране на програмата се заделя една и съща памет

Команди:

- echo – извежда подаден като аргумент текст на стандартния изход
- read – чете от стандартния си вход текст и го записва в променливи

- test – проверява дали дадено условие е изпълнено за някакви данни, подадени като параметри. Има различни типове опции, например за сравнение на числа, на символни низове, проверки на метаданни на файлове и т.н.
- if – условие за разклоняване – приема даден логически израз, който се оценява като истина или лъжа, и ако резултатът е истина, изпълнява даден блок код (последователност от инструкции)
- while – инструкция за повтаряне на някакво действие докато даден логически израз се оценява като истина
- for – инструкция за повтаряне на някакво действие докато дадена единица принадлежи на някакво множество

13.

**Тема 13**

Таблици за съответствието виртуална/реална памет.
Ефективна обработка на адресацията – MMU, TLB.

файлови дескриптори, номера на стандартните fd, пренасочване
филтри – cat, grep, cut, sort, wc, tr

Хардуерът, който отговаря за транслацията между виртуален и физически адрес на страницата (за управление на адресацията изобщо), се нарича MMU (Memory Management Unit). Инструментът, който използваме за оптимизиране на работата на MMU, се нарича TLB (Translation Lookaside Buffer). Модерните MMU обикновено разделят виртуалната памет на страници, всяка от които има размер, който е степен на 2 – обикновено няколко килобайта, но може и да са по-големи. Най-долните битове на адреса остават непроменени, горните битове на адреса са числата на виртуалната страница.

TLB е кеш, който хардуерът за управление на паметта използва, за да подобри скоростта при транслация на виртуални адреси. TLB има фиксиран брой слотове, съдържащи записи от таблицата със страници и таблицата със

сегменти. Записите от таблицата със страници се използват за преобразуване на виртуалните адреси във физически адреси, а записите от таблицата със сегменти – за преобразуване на виртуалните адреси в сегментни адреси.

Файлови дескриптори – това са цели положителни числа, които ОС асоциира с дадени файлове, върху които могат да се осъществяват входно-изходни операции

Номера на стандартните fd – 0 (stdin), 1 (stdout), 2 (stderr)

Пренасочването представлява изпращане на изходните данни, получени при изпълнението на даден процес, посредством тръба (FIFO) към стандартния вход на друг обект (|, <, >, >>, …)

Команди:

- cat – извежда на стандартния изход съдържанието на даден файл под формата на текст
- grep (global regular expression print) – приема като аргумент файл (или чете от стандартния вход) и търси съвпадение на някакъв шаблонен символен низ в текста
- cut – избира определено множество от колони в даден текст и извежда само тях (може да избира и определена последователност от символи), приема аргумент файл или чете от стандартния вход
- sort – сортира (прочетеното от стандартния вход или аргумент файл) по дадена колона и по различни критерии (нарастващ/намаляващ ред, лексикографско/числено сортиране и т.н.)
- wc – преброява в даден файл (или чете от стандартния вход) символите, байтовете, редовете, думите и т.н.
- tr – заменя срещането на даден символ с друг (чете от стандартния вход или приема аргумент файл), също може да изтрие съседни срещания на един и същи символ, или да изтрие всяко срещане на даден символ

14.

**Тема 14**

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

Опишете накратко стандарта RAID5. Какво е журнална файлова система?

Свързване и допускане до UNIX система – login.

Конзола – стандартен вход, стандартен изход, стандартна грешка.

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

Причини:

- Дефект в хардуера
- Софтуерна грешка
- Вирус (злонамерена атака)
- Грешка на потребителя
- Спиране на захранването

RAID5 е механизъм при организация на паметта, който предотвратява загуба на информация. При него се използват минимум 3 диска и се разделят на сектори. За всеки пореден сектор на някой от дисковете се пресмята сумата по модул 2 на този сектор от останалите дискове. Така, който и диск да бъде повреден, може да се възстанови чрез тази контролна сума.

Журнална файлова система – файлова система, при която незаписаните промени се запазват в правилния ред в структура от данни, наречена журнал.

Login – При влизане в UNIX система потребителят трябва да въведе своето потребителско име и парола в интерфейса на логин процеса. След което системата сравнява въведените данни със записите, където се съхранява информацията за потребителите и техните удостоверяващи данни. Ако въведените данни съвпадат със запис на потребителски акаунт, се дава достъп на потребителя до системата.

Стандартните I/O връзки в UNIX са три – stdin (0), stdout (1), stderr (2). По подразбиране stdin е свързан с клавиатурата, а stdout и stderr – с терминала.

Shell – след като сме се вписали успешно в системата, се задейства програма shell. Тази програма има много вариации, служи за посредник между крайния потребител и ОС. Интерпретира командите, които потребителят въвежда, търси за съответстващи им изпълними програми, и ги изпълнява.

Командата е инструкция, която потребителят дава на ОС, като параметрите служат за изменяне по някакъв начин на поведението на командата, за да се получат по-специфични резултати.

15.

**Тема 15**

Опишете разликата между синхронни и асинхронни входно-изходни операции.

Дайте примери за програми, при които се налага използването на асинхронен вход-изход.

Опишете с по едно-две изречения работата на следните системни извиквания в стандарта POSIX:

socket(), bind(), connect(), listen(), accept()

При синхронните входно-изходни операции процесът проверява дали ресурсът е свободен. Ако е, получава нужните данни, в противен случай се приспива. Събужда се едва когато ресурсът се освободи.

При асинхронните входно-изходни операции процесът, дори и да не получи правилно данните, не се приспива. Вместо това процесът може да свърши друга полезна работа. Това допринася за по-голяма гъвкавост на работата, но потребителят има грижата да отправи отново заявка за данните при несполучлива операция.

Програми, при които се налага използването на асинхронен вход-изход, са например уеб браузърите. Те приемат входни данни от мишката, клавиатурата, информация от интернет. Това са поне три различни канала на информация. Те трябва да могат да комуникират асинхронно по всеки един от тези канали и да реагират по подходящ начин.

Друг пример са сървърите, които обслужват множество клиенти. Те трябва да следят по кои комуникационни канали има готовност за входно-изходни операции и кои са пасивни.

Системни извиквания:

- socket() – инициализира специален файл от тип socket, който служи за връзка по мрежата с процеси, които не е нужно да са деца на текущия, а могат да са всякакви
- bind() – задава име на настоящия socket, като по този начин той става откриваем в мрежата
- connect() – изпраща заявка за свързване от страна на клиентски socket към сървърен socket, който е намерен в мрежата
- listen() – сървърният socket започва да следи за постъпили заявки за свързване по мрежата
- accept() – сървърният socket приема заявка за свързване и осъществява двустранна връзка за комуникация с ответния процес.

16.

**Тема 16**

Опишете понятието „пространство на имената“ (VFS).
Структура, обекти и техните атрибути във VFS за ОС Linux.
Основни функции, които обслужва пространството на имената.

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

Пространството на имената е съвкупността от всички дълготрайни обекти във файловата система. Нарича се още виртуална файлова система (Virtual File System – VFS).

В ОС Linux обектите са подредени в кореново дърво, където междинните върхове представляват директории, а листата – файлове. Към това дърво можем да добавяме други устройства с mount/unmount. Всеки от тези обекти има атрибути като тип, име, име на собственик, група, разрешения, дата на създаване/модификация/промяна/достъп.

Пространството на имената предлага ефективна изолация на процесите, което помага за по-гъвкав контрол върху тях.

Задачата за читателите и писателите представлява ситуация, в която има два типа процеси – читатели и писатели, и някаква структура от данни, която се ползва и от двете страни. Читателите могат да четат от нея, а писателите могат да я модифицират. Условието е читателите да могат да влизат едновременно в „стаята“ и да четат от тези данни, а писателят да може да влиза само когато стаята е празна и да получава ексклузивни права над нея докато модифицира данните. Също така, не трябва да се допуска гладуване при някоя от страните, или да има deadlock.

Реализация чрез семафори:

Semaphores: mutex, room_empty, barrier

int count = 0

barrier.init(1)

mutex.init(1)

room_empty.init(1)

WRITERS

barrier.wait()

room_empty.wait()

{write}

barrier.signal()

roomEmpty.signal()

READERS

barrier.wait()

barrier.signal()

mutex.wait()

++count

if count == 1 :

room_empty.wait()

mutex.signal()

{read}

mutex.wait()

--count

if count == 0:

room_empty.signal()

mutex.signal()

17.

**Тема 17**

Опишете какви атрибути имат файловете в съвременна файлова система, реализирана върху блочно устройство (block device).

Опишете накратко реализацията и целта на следните инструменти:

(а) отлагане на записа, алгоритъм на асансьора.

(б) поддържане на буфери (кеширане) на файловата система.

Опишете как се изгражда комуникационен канал (connection) между процес-сървер и процес-клиент със следните системни извиквания в стандарта POSIX:

socket(), bind(), connect(), listen(), accept()

Файловете имат следните атрибути: име на файла, име на собственика, група, размер, тип (f, d, s, p, b, l), разрешения (права) за достъп, време на създаване/модификация/промяна/достъп. В съвременните ос всеки файл/директория се изразява чрез наредената двойка <име, inode>.

Кешът на файловата система съдържа данни, които наскоро са прочетени от диска, така следващите заявки могат да получат данните от кеша вместо да четат отново от диска.

18.

**Тема 18**

Опишете накратко основните комуникационни канали в ОС Linux.
Кои канали използват пространството на имената и кои не го правят?

Опишете какви изисквания удовлетворява съвременна файлова система, реализирана върху блочно устройство (block device).
Опишете предназначението на журнала на файловата система.

Основните комуникационни канали в ОС Linux са следните:

- FIFO (неименувана тръба): Реализира се с командата pipe. Служи за връзка между процес и негови child процеси (транзитивно). Не използва пространството на имената
- Именувана тръба: Реализира се с командата mkfifo. Представлява специален файл, който има поведение на тръба и може да бъде достъпен от различни процеси в системата. Използва пространството на имената.
- Връзка процес-файл: Инициализира се с командата open. Канал, който се създава между процес и даден файл от файловата система, от единия край се изгражда от ядрото, а от другия край е съответният файл, върху който се изпълняват входно-изходни операции. Използва пространството на имената, понеже само така файлът може да бъде локализиран.
- Socket: Инициализира се с командата socket. Служи за връзка по мрежата между текущия процес и друг процес, който може да има произволен произход. Асоциира се с пространството на имената от сървърната страна, защото само така може да бъде намерен от клиентския процес.

Какви изисквания удовлетворява съвременна файлова система, реализирана върху блочно устройство: Трябва да бъде надеждна, ефективна, да поддържа многопотребителска среда. Също така трябва да осигурява сигурност на данните.