Технически Университет – София

Факултет Приложна Математика и Информатика

Катедра Информатика

КУРСОВА РАБОТА

Тема: Тема за реферат 15

Имена на студента: Станислав Бисеров Стоянов

Факултетен номер, група: 471218066, 76 група

Съдържание:

1. Каква е основната разлика между блоковия специален файл и символния специален файл?
2. Представете си, че имаме система за съобщения, която използва пощенски кутии. Когато изпращаме съобщения към пълна пощенска кутия или получаваме в празна такава, процесът не се блокира. Вместо това, се изпраща грешка (error code). Процесът отговаря на грешката като пробва да се изпълни отново и отново докато успее. Цялото това действие води ли до условие на състезание? (race condition)
3. Виртуалната памет осигурява механизъм за изолиране на един процес от друг. Какви трудности при управлението на паметта биха били включени при опита две операционни системи да работят едновременно (concurrently)? Как тези затруднения ще бъдат адресирани?
4. Визуализация

1. Каква е основната разлика между блоковия специален файл и символния специален файл?

Когато една програма чете и записва данни във файл, заявките се изпращат към kernel драйвър. Ако файлът е обикновен (regular), данните се обработват от драйвъра на файловата система и типично се съхраняват в различни частти на твърдия диск или върху друг външен носител на информация. Данните, които са прочетени от този файл на предишен етап са били записани отново на същото място. Същият принцип е валиден и когато данните се четат или записват върху файл на дисковото устройство. Като този път заявката се обработва от драйвъра, който отговаря за самото устройство. Всъщност какво се случва с данните е единствено отговорност на устройството и драйвъра, който ползва.

- Блоковите устройства (*block devices или block special files*) обикновено наподобяват много обикновите файлове: те са масив от байтове и стойността, която е прочетена от тях се намира на същото място, където последно е било записано нещо. Данните от блоковото устройство могат да бъдат кеширани в паметта и прочетени обратно от там, като също така записванията могат да бъдат буферирани. Много важна характеристика на този тип файлове е, че те могат да бъдат променяни (seekable), тоест всяка една позиция вътрешно в тях може да бъде променена от дадено приложение. От къде идва името на този тип файлове? Всъщност наименованието идва от факта, че хардуерът обикновено чете и записва цели блокове данни, например сектор на твърдия диск. Друга важна характеристика за тези файлове е, че те съдържат в себе си номерирани блокове, всеки, от които може да бъде прочетен или записан независимо от всички останали. Блоковите устройства са способни да буферират резултата (output) и да съхраняват данните за по-късен етап. Пример за такива устройства са: твърдият диск и паметта.
- Символните устройства (character devices или character special files) наподобяват много на конекторите (pipes) и серийните портове. Писането и четенето в тях се случва незабавно. Писането на байт в такъв тип файл може лесно да бъде визуализирано на екрана, подадено като резултат на сериен порт или просто конвертирано в звуков сигнал. Четенето на байт може да накара серийния порт да изчака за вход или просто може да върне рандъм байт. От къде идва името на този

тип файлове? При символните устройства всеки един символ се обработва индивидуално. Драйвърите на тези устройства са специални файлове, които позволяват на ОС да комуникира с входно/изходните устройства. Пример за такива устройства са: клавиатурата, мишката, мониторът, аудио и видео картите.

- Какви са основните разлики между двата вида устройства?

При блоковите устройства е възможно да се търси във всеки един блок и да се започне четене или писане, докато при символните устройства това не е възможно. Също така основната разлика е и в това какво количество данни може да бъде прочетено или записано от ОС и хардуера.

2. Представете си, че имаме система за съобщения, която използва пощенски кутии. Когато изпращаме съобщения към пълна пощенска кутия или получаваме в празна такава, процесът не се блокира. Вместо това, се изпраща грешка (error code). Процесът отговаря на грешката като пробва да се изпълни отново и отново докато успее. Цялото това действие води ли до условие на състезание? (race condition)

Ако пощенската кутия поддържа атомарност (atomicity – атомарна операция е тази, която се изпълнява изцяло, без да бъде прекъсвана от ОС) при получаването на имейлите, няма да възникне условие на състезание (race condition). Но нека си представим, че пощенската кутия не използва никакъв заключващ механизъм. Ако по едно и също време се опитаме да прочетем съобщение и ново съобщение пристигне в пощенската кутия, то тогава можем да прочетем незапълнено съобщение, защото то все още не е пристигнало. В най-добрият случай ще получим част от съобщението, но обикновено настъпва грешка заради някакъв невалиден пойнтър (pointer). Също така понеже комуникацията е в двете посоки и всеки процес зависи от останалия, race condition няма никога да настъпи. Единствено ще се получи безкрайно изчакване (цикъл) до получаване на успешен изход. Какво всъщност представлява условието на състезанието (race condition)? Представлява ситуация, при която няколко процеса конкурентно манипулират данни, които са общи, поделени между конкурентните процеси. Крайната стойност на поделените данни зависи от начина, по който процесите се планират върху процесора. За да се предотврати това

състояние, конкурентните процеси трябва да бъдат синхронизирани посредством заключващи механизми.

3. Виртуалната памет осигурява механизъм за изолиране на един процес от друг. Какви трудности при управлението на паметта биха били включени при опита две операционни системи да работят едновременно (concurrently)? Как тези затруднения ще бъдат адресирани?

Първоначално нека дефинираме понянието виртуална памет. Виртуалната памет е множество отделни блокове от последователни адреси (страници). Най-разпространеният размер на една страница е 4KB. Тези страници съществуват независимо една от друга. Първоначално идеята е била необходимите в момента страници да са в реалната памет, а останалите на диска. Но в наши дни реалното адресно пространство е колкото виртуалното. Виртуалната памет е адресируема памет. Адресът се нарича виртуален, но може да бъде срещнат в различни литературни източници и под наименованията логически, математически и др. Относно отговора на въпроса, той се отнася до един от основните аспекти на виртуалната поддръжка на машината. Основното препятствие е как да се постигне максимална производителност единствено с необходимата памет заделена за операционната система. Проблемът е в това как бързо да се премине към използване на втората операционна система и как да се използва правилно TLB-буферът, който притежава много по-голяма степен на асоциативност, отколкото другите типове кеш-памети. Трябва да се имат предвид несполуките, които може да се получат при обръщение към този буфер, тъй като те водят до генериране на прекъсвания. Обикновено, стандартен подход е да заделим някакъв брой TLB записи за всяко ядро и да подсигурим, че тези ядра ще работят правилно с виртуалната памет. Но за жалост, хардуерът и в частност Интел архитектурите поправят настъпилите проблеми с TLB буфера без нашето знание. Затова е съществено важно да обработим сами тези проблеми и да знаем, че намирането на физическо място на търсената информация изисква ресурси и време. Това е причината да се ползва този адресен кеш-буфер, който има основно предимство, че е достъчно бърз.

4. Визуализация

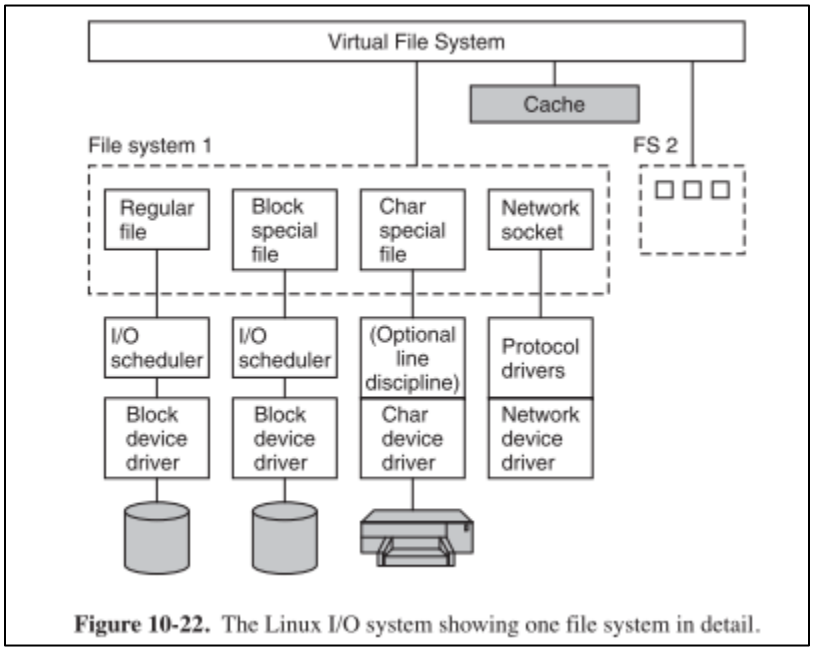

Figure 1- Място на блоковия и символния тип файл във виртуалната файлова система

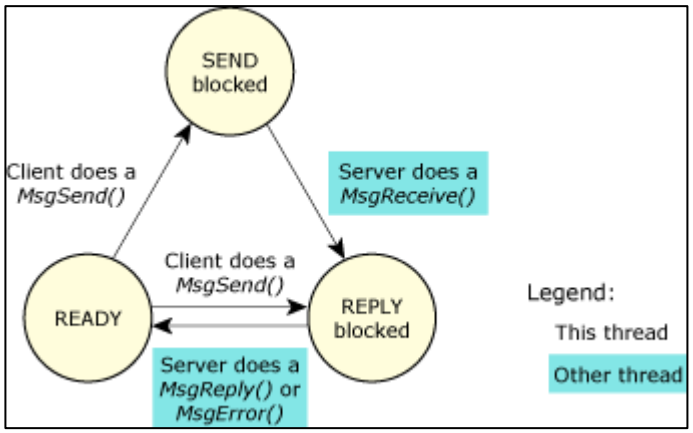

Figure 2- Изпращане и получаване на съобщение с изчакване