РАБОТА С ВНЕШНИМИ УСТРОЙСТВАМИ LINUX

Классы устройств и модулей

- Linux разделяет все устройства на три основных типа. Каждый модуль, как правило, реализует функциональность одного из этих типов и отсюда может классифицироваться как символьный модуль, блочный модуль или сетевой модуль. Ниже приводятся краткие описания этих трех классов:
- **Символьные устройства**
- Символьное устройство - устройство, которое поставляет данные как поток байтов. Драйвер символьного устройства отвечает за реализацию поддержки такого поведения. Такие драйверы реализуют как минимум четыре системных вызова: open, close, write и read. Примеры таких устройств - текстовая консоль (/dev/console) и последовательный порт (/dev/ttyS*).

Классы устройств и модулей

- К символьным устройствам можно обращаться посредством элементов файловой системы. Важное отличие обычных файлов от символьных устройств заключается в том, что программа может перемещаться по файлам вперёд и назад, в то время как символьные устройства это лишь каналы передачи данных, доступ к которым осуществляется в заданном порядке.
- **Блочные устройства**
- К блочным устройствам так же можно обращаться посредством элементов файловой системы в каталоге /dev. Примером блочного устройства может служить жёсткий диск. Обмен данными с блочным устройством производится порциями байт - блоками.

Классы устройств и модулей

- В большинстве Linux-систем размер одного блока равен 1 килобайту или другому числу, являющемуся степенью числа 2. Linux позволяет приложениям обращаться к блочному устройству так же как к символьному он разрешает передачу любого числа байт в блоке. В результате, все различие между блочными и символьными устройствами сводится к внутреннему представлению данных в ядре. Драйвер блочного устройства реализует точно такой же интерфейс с ядром, что и драйвер символьного устройства, но дополнительно реализуется еще и блочно-ориентированный интерфейс, который "невидим" для пользователя или приложения, которые открывают доступ к блочному устройству посредством псевдофайловой системы /dev.

Классы устройств и модулей

- **Сетевые интерфейсы**
- Любой сетевой обмен выполняется через сетевой интерфейс, то есть устройство, которое способно обмениваться данными с другими узлами сети. Как правило, это аппаратное устройство, но возможна и программная реализация сетевого устройства, например петлевое устройство loopback. Сетевой интерфейс отвечает за передачу и получение пакетов данных, которыми управляет сетевая подсистема ядра, не зная о том, к каким соединениям эти пакеты принадлежат. Не смотря на то, что соединения по протоколам Telnet и FTP используют один и тот же сетевой интерфейс, само устройство не различает эти соединения, оно "видит" только пакеты данных.

Классы устройств и модулей

- **Файловые системы**
- Кроме драйверов устройств, самым важным классом модулей в Linux являются файловые системы. Тип файловой системы обусловливает способ организации информации на блочном устройстве, это программный драйвер, потому что он отображает структуры данных нижнего уровня на структуры данных верхнего уровня. Файловая система определяет какой длины могут быть имена файлов и какая информация о каждом из файлов должна храниться. Файловая система должна реализовать самый нижний уровень системных вызовов для доступа к каталогам и файлам, путем отображения их имён (и иной информации, такой как права доступа и пр.) в структуры данных, которые записываются в блоки данных.

Работа со временем

- Часто требуется засекать время выполнения программы, это можно сделать с помощью разных методов, но все они базируются на прерываниях от таймера. Прерывания от таймера представляют собой механизм, используемый ядром для получения требуемых интервалов времени. Прерывания представляют собой асинхронные события, которые обычно генерируются каким-либо внешним физическим устройством. При этом CPU прерывает исполнение своей текущей задачи, и начинает исполнять специальный код обработчика прерывания.

Работа со временем

- Прерывания таймера генерируются специальным системным электронным компонентом через равные заданные промежутки времени. Значение интервала времени устанавливается ядром, измеряется в Гц, является архитектурно-зависимым, и определено в заголовочном файле <linux/param.h> макроопределением HZ. Современный Linux, на большинстве платформ, использует прерывания с частотой 100 Гц. Некоторый платформы используют значение 1024 Гц.
- Каждый раз, когда возникает прерывание от таймера, инкрементируется значение переменной jiffies. jiffies инициализируется нулем при загрузке системы, и, таким образом, показывает количество тиков таймера с момента включения компьютера.

Работа со временем

- Можно изменить значение интервала таймера изменением макроопределения HZ, это можно использовать при решении задач реального времени, но это приведет к увеличению общей доли времени, которое тратится на обработку прерывания таймера.
- **Процессоро-зависимые регистры**
- Если необходимо измерить очень короткие интервалы времени, или требуется высокая точность измерения, можно использовать платформо-зависимые ресурсы, выбирая точность против портируемости результирующего кода.
- Большинство современных CPU содержат в себе счетчик, значение которого инкрементируется с каждым тактом процессора. Такой счетчик может быть использован для точного измерения интервалов времени.

Работа со временем

- Учитывая свойственную большинству систем непредсказуемость времени исполнения инструкций (по причинам диспетчеризации, вероятностных предсказаний переходов, и различных уровней кэша памяти), такой счетчик процессорных тиков представляет собой единственно надежный способ выполнения задач точного измерения времени.
- Наиболее известным регистром счетчиком является регистр TSC (timestamp counter), представленный в семействе процессоров x86 начиная с процессора Pentium. Это 64-битный регистр, который считает такты CPU. Он может быть прочитан как из пространства ядра, так и из пространства пользователя.

Работа со временем

- Включив заголовочный файл <asm/msr.h> (machine-specific registers), можно использовать следующие макросы:
- `rdtsc(low,high);`
- `rdtscl(low);`
- С помощью первого макроса можно прочитать 64-битное значение счетчика в две 32-битовые переменные. Второй макрос читает младшую половину регистра в 32-х битовую переменную, и является достаточным в большинстве случаев. Например, на 500 МГц процессоре переполнение 32-х битового счетчика будет происходить каждые 8.5 секунд.

Работа со временем

- Заголовки ядра включают и архитектурно-независимую функцию, скрывающую существующие различия реализации, и которую можно использовать вместо rdtsc(). Она называется get_cycles() (определена в <asm/timex.h>). Её прототип:
- `#include <linux/timex.h>`
- `cycles_t get_cycles( void );`
- Эта функция определена для любой платформы, и она всегда возвращает нулевое значение на платформах, которые не имеют реализации регистра счётчика циклов. Тип cycles_t является соответствующим целочисленным типом без знака для хранения считанного значения.

Работа со временем

- Такой выбор емкости означает, например, что на процессорах Pentium функция get_cycles() возвратит только младшие 32 бит реального счетчика процессорных тиков. Это позволит избежать многорегистровых операций, и не воспрепятствует цели наиболее частого использования счетчика - измерение коротких интервалов времени.
- **Примечание:** Нулевое значение, возвращаемое get_cycles() на платформах, не предоставляющих соответствующей реализации, делает возможным обеспечить переносимость между аппаратными платформами тщательно прописанного кода (там, где это есть, используется get_cycles(), а там, где этой возможности нет, тот же код реализуется, опираясь на последовательность системных тиков).

Работа со временем

- **Функция do_gettimeofday()**
- Если драйверу необходимо текущее время, можно воспользоваться функцией do_gettimeofday(). Она заполняет экземпляр структуры timeval, переданный в функцию по указателю, с использованием значений секунд и микросекунд. Для пространства пользователя существует аналогичный системный вызов gettimeofday(). Прототип для функции do_gettimeofday() выглядит следующим образом:
- `#include <linux/time.h>`
- `void do_gettimeofday(struct timeval *tv);`
- Функция do_gettimeofday() имеет "разрешение в пределах микросекунды" для многих архитектурах.

Работа со временем

- **Системный вызов times()**
- Каждый процесс может пребывать в двух фазах: системной (внутри тела системного вызова - его выполняет для нас ядро операционной системы) и пользовательской (внутри кода самой программы). Время, затраченное процессом в каждой фазе, может быть измеряно системным вызовом times(). Кроме того, этот вызов позволяет узнать суммарное время, затраченное порожденными процессами [2].
- Системный вызов заполняет структуру
- `struct tms { clock_t tms_utime; clock_t tms_stime; clock_t tms_cutime; clock_t tms_cstime; };`

Работа со временем

- Все времена измеряются в "тиках" - некоторых долях секунды. Число тиков в секунде можно узнать таким системным вызовом:
- `clock_t HZ = sysconf(_SC_CLK_TCK);`
- Поля структуры содержат:
- tms_utime(tms_cutime) - время, затраченное вызывающим(порожденным) процессом в пользовательской фазе,
- tms_stime(tms_cstime) - время, затраченное вызывающим(порожденным) процессом в системной фазе.
- real_time - время, соответствующее астрономическому времени системы. Если нужно засечь интервал времени выполнения программы, то имеет смысл мерять только их разность.

Работа с внешними устройствами в ОС Linux

- **Метод ioctl()**
- Данный метод работает через файловый дескриптор. В него передается целое число, символизирующее запрашиваемую команду, и другие аргументы, обычно – указатель. Преимущества этого метода перед другими: если необходимо выполнять обработку полученных из драйвера данных прежде, чем они будут выведены на экран, то получение данных в бинарной форме более эффективна, нежели их извлечение из текстового файла /proc; получение данных через ioctl() не связано со страничной фрагментацией; является возможность оставления команд получающих отладочную информацию после отладки.

Работа с внешними устройствами в ОС Linux

- Вызов функции ioctl() в пользовательском процессе определяется следующим прототипом :
- `int ioctl(int fd, int cmd, ...);`
- Действительный тип третьего аргумента зависит от передаваемой в процедуру команды, которая определяется вторым аргументом. Некоторые команды не имеют аргументов, некоторые используют целый аргумент, а некоторые берут в качестве аргумента указатель на структуру данных.
- С другой стороны, функция ioctl() драйвера получает аргументы согласно следующему объявлению:
- `int (*ioctl) (struct inode *inode, struct file *filp, unsigned int cmd, unsigned long arg);`

Работа с внешними устройствами в ОС Linux

- Указатели inode и filp представляют собой значения соответствующие файловому дескриптору fd, переданному пользовательским процессом, и полностью совпадают с параметрами, передаваемыми в системный вызов open(). Аргумент cmd передается от пользователя неизменным, а необязательный аргумент arg передается в форме unsigned long, что может соответствовать как целому значению, так и указателю. Если вызывающая программа не передает третий аргумент, то значение arg имеет неопределенное значение.

Работа с внешними устройствами в ОС Linux

- **Управление устройством через esc-последовательности**
- Управление некоторыми устройствами удачнее реализуется через запись управляющей последовательности в само устройство. Такая техника, например, используется для драйвера консоли, в который передаются, так называемые esc-последовательности, используемые для перемещения курсора, изменения цвета, и выполнения других конфигурационных задач. Выгода такого способа управления заключается в том, что пользователи, имеющие права на запись в устройство могут участвовать в управлении им не приобретая специальных привилегий, и не используя специальных программ, реализующих ioctl() вызовы.

Работа с внешними устройствами в ОС Linux

- Например, программа setterm управляет конфигурацией консоли передачей esc-последовательностей. Дополнительным преимуществом такого управления является простота удаленного управления устройством. Управляющая программа и управляемое устройство могут размещаться на разных компьютерах, т.к. управление может осуществляться простым перенаправлением потока данных.
- Неприятной стороной такого управления устройством являются дополнительные ограничения по управлению устройством. Например, управляющие последовательности не должны оказаться внутри потока данных передаваемых в устройство. Это справедливо и для tty.

Работа с внешними устройствами в ОС Linux

- Несмотря на то, что текстовый дисплей предназначен для отображения только символов ASCII, иногда, в потоке передаваемых данных, могут оказаться управляющие символы, влияющие на настройку консоли.
- **Использование файловой системы /proc**
- Файловая система /proc представляет собой специальную, программно-реализованную файловую систему, которая связана с функциями ядра, которые генерируют содержание файла во время чтения. Например /proc/modules, который возвращает список загруженных, в данный момент модулей и /proc/meminfo, который содержит статистику использования памяти. Файловая система /proc широко используется в Linux-системе.

Работа с внешними устройствами в ОС Linux

- Многие утилиты современных Linux-дистрибутивов, такие как ps, top и uptime получают свою информацию из /proc. Некоторые драйвера устройств, также используют /proc для передачи информации в пространство пользователя. Файловая система /proc является динамической системой, и модуль может добавлять и удалять файловый элемент из этой системы во время своей работы.
- Метод использования файловой системы /proc похож на работу с драйверами устройства: создается структура со всей информацией, необходимой для /proc файла, включая указатели на любые функции драйвера. Размер файла равен нулю потому что содержание файла генерируется ядром, файл нельзя запустить. Время модификации данного файла – текущее время.

Работа с устройствами PCI и USB в ОС Linux

- Для программирования задач управления данными устройствами требуются соответствующие библиотеки. Рассмотрим пакеты, в которые они входят.
- **Пакет PCI Utilities-3.1.8**
- Пакет PCI Utilities является набором программ, предназначенным для выдачи списка устройств PCI, проверки состояния устройств и настройки их конфигурационных регистров.
- **Информация о пакете.** Загрузка: ftp://atrey.karlin.mff.cuni.cz/pub/linux/pci//pciutils-3.1.8.tar.gz

Работа с устройствами PCI и USB в ОС Linux

- Размер загружаемого пакета: 372 KB
- Оценочный размер требуемого дискового пространства: 3,5 MB
- Замечания для пользователей: http://wiki.linuxfromscratch.org/blfs/wiki/pciutils
- **Установка пакета PCI Utilities**
- Установите пакет PCI Utilities с помощью следующих команд:
- make PREFIX=/usr ZLIB=no В этом пакете набор тестов отсутствует.

Работа с устройствами PCI и USB в ОС Linux

- Теперь в роли пользователя root выполните:
- make PREFIX=/usr install Некоторые пакеты требуют статической библиотеки PCI. Чтобы установить библиотеку и заголовки, выполните в роли пользователя root следующую команду:
- make PREFIX=/usr install-lib **Пояснение команды**
- ZLIB=no: Этот параметр предотвращает сжатие файла pci.ids, который необходимо для других приложений, например, HAL-0.5.14.

Работа с устройствами PCI и USB в ОС Linux

- **Конфигурирование пакета PCI Utilities**
- Файл данных pci.ids постоянно обновляется. Чтобы получить текущую версию этого файла, выполните в роли пользователя root команду **update-pciids**. Для этой программы требуется скрипт Which-2.20. Для этой программы требуется скрипт Which-2.20 или программа, которая ищет cURL-7.22.0. Для этой программы требуется скрипт Which-2.20 или программа, которая ищет cURL-7.22.0, Wget-1.13.4. Для этой программы требуется скрипт Which-2.20 или программа, которая ищет cURL-7.22.0, Wget-1.13.4 или Lynx-2.8.7rel.2, используемые для загрузки самого нового текущего файла, а затем замены им файла, находящегося в /usr/share.

Работа с устройствами PCI и USB в ОС Linux

- **Установленные программы:** lspci, setpci и update-pciids
- **Установленные библиотеки:** libpci.a
- **Установленные директории:** /usr/include/pci
- **Краткое описание:**
- **Lspci** - это утилита для отображения информации о всех шинах PCI, имеющихся в системе, и всех устройств, подключенных к ним.
- **Setpci -** это утилита для запросов к устройствам PCI и их конфигурирования.

Работа с устройствами PCI и USB в ОС Linux

- **Пример:** setpci –v -s 07:00.0 F4.B=FF – установить максимальную яркость монитора сигналом с видеокарты.
- Ключи: -v verbose (подробный отчет), -s site (domain:bus:slot.func), -D demo (не выполнять).
- Команды: регистр.тип=значение. Типы: Byte, Word, Long.
- Получить список регистров: setpci - -dumpregs
- Возможна сборка программы управления конкретным устройством из исходных кодов. Пример кода – здесь.
- **update-pciids -**выдает текущую версию списка PCI ID. Требует Which-2.20выдает текущую версию списка PCI ID. Требует Which-2.20, cURL-7.22.0выдает текущую версию списка PCI ID. Требует Which-2.20, cURL-7.22.0, Wget-1.13.4выдает текущую версию списка PCI ID. Требует Which-2.20, cURL-7.22.0, Wget-1.13.4 или Lynx-2.8.7rel.2

Работа с устройствами PCI и USB в ОС Linux

- **libpci.a** - является статической библиотекой, которая позволяет приложениям получать доступ к подсистеме PCI.
- **Пакет usbutils-004**
- В пакете usbutils находится утилита, используемая для отображения информации о шинах USB, имеющихся в системе, и устройствах, подключенных к ним.
- **Информация о пакете.** Загрузка (HTTP): http://ftp.de.debian.org/debian/pool/main/u/usbutils/usbutils_004.orig.tar.bz2z

Работа с устройствами PCI и USB в ОС Linux

- Размер загружаемого пакета: 472 KB
- Оценочный размер требуемого дискового пространства: 4,5 MB
- **Зависимости пакета usbutils. Обязательные:**
- libusb-1.0.8
- Замечания для пользователей: http://wiki.linuxfromscratch.org/blfs/wiki/usbutils
- **Установка пакета usbutils**
- Установите пакет usbutils с помощью следующих команд:

Работа с устройствами PCI и USB в ОС Linux

- ./configure --prefix=/usr --disable-zlib &&make В этом пакете набор тестов отсутствует.
- Теперь в роли пользователя root выполните:
- make install &&mv -v /usr/sbin/update-usbids.sh /usr/sbin/update-usbids **Пояснение команды**
- --disable-zlib: Этот параметр указывает пакету usbutils не устанавливать файл usb.ids ни в сжатом, ни в распакованном виде.

Работа с устройствами PCI и USB в ОС Linux

- **Конфигурирование пакета usbutils**
- Файл данных usb.ids постоянно обновляется. Чтобы получить текущую версию этого файла, выполните в роли пользователя root команду update-usbids. Для этой программы требуется скрипт Which-2.20Файл данных usb.ids постоянно обновляется. Чтобы получить текущую версию этого файла, выполните в роли пользователя root команду update-usbids. Для этой программы требуется скрипт Which-2.20 или программа, которая ищет cURL-7.22.0Файл данных usb.ids постоянно обновляется. Чтобы получить текущую версию этого файла, выполните в роли пользователя root команду update-usbids. Для этой программы требуется скрипт Which-2.20 или

Работа с устройствами PCI и USB в ОС Linux

- **Описание пакета**
- **Установленные программы:** lsusb, update-usbids и usb-devices
- **Установленные библиотеки:** Нет
- **Установленные директории:** Нет
- **Краткое описание**
- **Lsusb -** это утилита для отображения информации о всех шинах USB, имеющихся в системе, и всех устройств, подключенных к ним.

Работа с устройствами PCI и USB в ОС Linux

- **update-usbids -** выдает текущую версию списка USB ID. Требует Which-2.20выдает текущую версию списка USB ID. Требует Which-2.20, cURL-7.22.0 выдает текущую версию списка USB ID. Требует Which-2.20, cURL-7.22.0, Wget-1.13.4выдает текущую версию списка USB ID. Требует Which-2.20, cURL-7.22.0, Wget-1.13.4> или Lynx-2.8.7rel.2
- **usb-devices -** это скрипт командной строки, который отображает подробную информацию о шинах USB и устройствах, подключенных к ним. Он используется, если в вашей системе недоступны /proc/bus/usb/devices.
- **Пакет libusb-1.0.8**
- В пакете libusb находится библиотека, используемая

Работа с устройствами PCI и USB в ОС Linux

- **Информация о пакете**
- Загрузка (HTTP): http://downloads.sourceforge.net/libusb/libusb-1.0.8.tar.bz2
- Загрузка (FTP): ftp://anduin.linuxfromscratch.org/BLFS/svn/l/libusb-1.0.8.tar.bz2
- Размер загружаемого пакета: 331 KB
- Оценочный размер требуемого дискового пространства: 5 MB

Работа с устройствами PCI и USB в ОС Linux

- **Зависимости пакета libusb**
- **Необязательные (необходимы для сборки документации по API) -** Doxygen-1.7.5
- Замечания для пользователей: http://wiki.linuxfromscratch.org/blfs/wiki/libusb
- **Установка пакета libusb**
- Установите пакет libusb с помощью следующих команд:
- ./configure --prefix=/usr &&make

Работа с устройствами PCI и USB в ОС Linux

- Если установлен пакет Doxygen и вы хотите собрать документацию API, введите следующую команду:
- make -C doc docs В этом пакете набор тестов отсутствует.
- Теперь в роли пользователя root выполните:
- make install Если вы собрали документацию API, установите ее в роли пользователя root с помощью следующих команд:
- install -v -d -m755 /usr/share/doc/libusb-1.0.8/apidocs &&install -v -m644 doc/html/* \ /usr/share/doc/libusb-1.0.8/apidocs

Работа с устройствами PCI и USB в ОС Linux

- **Конфигурирование пакета Libusb**
- Чтобы получить доступ к "настоящим" устройствам USB (тем, которые драйвером запоминающих устройств не определяются как диск), в ядре должна быть соответствующая поддержка. Проверьте конфигурацию вашего ядра в меню Device Drivers => USB support => Support for Host-side USB (Драйвера устройств => Поддержка USB => Хост-поддержка USB). Там же вы можете выбрать для устройства USB любой аппаратный драйвер.

Работа с устройствами PCI и USB в ОС Linux

- **Описание пакета**
- **Установленные программы:** Нет
- **Установленные библиотеки:** libusb-1.0.{so,a}
- **Установленные директории:** /usr/include/libusb-1.0 и /usr/share/doc/libusb-1.0.8
- **Краткое описание**
- libusb-1.0.{so,a} - библиотеки, в которой находятся функции C для доступа к аппаратным возможностям USB
- Пример кода – здесь.

ПРИМЕРЫ

- **Опрос процессора**
- Задание: необходимо опросить процессор и вывести на экран информацию о его частоте в мегагерцах, названии модели, идентификаторе производителя и размере кэша. Вывести на экран время выполнения программы.
- Краткие теоретические сведения:
- Для опроса процессора удобнее всего воспользоваться файловой системой /proc описание которой приведено в пункте 4. Содержащаяся в /proc информация доступна для чтения человеку, но также ее можно извлечь программным путем с помощью парсинга. В данном случае необходимо получить информацию о процессоре, запустим /proc/cpuinfo командой #cat /proc/cpuinfo

ПРИМЕРЫ

- на экране получим следующее сообщение:

```
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 5
model name      : Pentium II (Deschutes)
stepping        : 2
cpu MHz         : 400.913520
cache size      : 512 KB
fdiv_bug        : no
hlt_bug         : no
sep_bug         : no
f00f_bug        : no
coma_bug        : no
```

ПРИМЕРЫ

- fpu : yes
- fpu_exception : yes
- cpuid level : 2
- wp : yes
- flags : fpu vme de pse tsc msr pae mce cx8 apic sep
- mtrr pge mca cmov pat pse36 mmx fxsr
- bogomips : 399.77
- Наилучший путь для получения информации из этого вывода – запись из файла в буфер и далее парсить в память используя функцию sscanf. Нужно помнить, что имена, семантика и формат выводимых данных может измениться в новых версиях ядра Linux. Поэтому при запуске программ нужно проверять содержание /proc.

ПРИМЕРЫ

- Методические указания
- 1) Перед началом работы выполнить команду
- #cat /proc/cpuinfo и убедиться в названии полей для тактовой частоты, идентификатора производителя и т.д.
- 2) В случае если используется текстовый буфер, нужно делать проверку на его заполнение.
- 3) В конце символьной строки нужно добавлять значение конца строки \0
- 4) Для работы с полученной информацией можно воспользоваться стандартными текстовыми функциями sscanf(), strstr() и т.д. Описание их можно найти по команде man.

ПРИМЕРЫ

- 5) Значение тактовой частоты процессора дробное, поэтому при выводе информации на экран нужно указать тип данных float с 3-4 знаками после запятой.
- 6) Засечь время выполнения программы можно любым из описанных выше способов.
- Код программы доступен здесь.

ПРИМЕРЫ

- **Опрос и управление модемом (последовательный порт)**
- Задание: Необходимо сконфигурировать последовательный порт, опросить модем, послать ему последовательность АТ-команд и вывести результат на экран. Вывести на экран время выполнения программы.
- Краткие теоретические сведения:
- Последовательная передача подразумевает передачу данных посылкой в один бит и применяется в большинстве сетевых устройств, клавиатурах, модемах и терминалах. Стандартный электрический интерфейс для последовательной передачи данных - RS-232.

ПРИМЕРЫ

- Linux предоставляет доступ к последовательным портам через файлы устройств. Для доступа к последовательному порту необходимо открыть соответствующий файл устройства. Каждый последовательный порт в системе Linux имеет несколько файлов устройств (файлы в каталоге /dev/ttyS0, /dev/ttyS1) Для открытия используется функция open():
- **`fd = open("/dev/ttyf1", O_RDWR | O_NOCTTY | O_NDELAY);`**
- При открытии устанавливаются два дополнительных флага вместе с режимом чтение/запись: O_NOCTTY - программа не является управляющим терминалом для этого порта.

ПРИМЕРЫ

- O_NDELAY - программа не следит за состоянием сигнала DCD (то есть что другой конец линии запущен). Если не установить этот флаг, то процесс не будет работать до тех пор пока на линии DCD не появится уровень space (off).
- Для записи данных в порт используется системный вызов write()
- `n = write(fd, "ATZ\r", 4);`
- Функция write возвращает количество посланных байт или -1 при возникновении ошибки.
- В случае, если идет работа с портом в режиме неструктурированного посимвольного обмена, каждый системный вызов read() будет каждый раз возвращать число прочитанных символов в буфер ввода.

ПРИМЕРЫ

- Если в текущий момент нет символов доступных для чтения, вызов будет блокироваться до тех пор, пока не появятся символы для чтения, или закончится счетчик таймаута, или обнаружится какая-нибудь ошибка. Функцию read можно выполнить так, что она вернет управление немедленно.
- `fcntl(fd, F_SETFL, FNDELAY);`
- Опция FNDELAY указывает функции read возвращать 0 если нет символов доступных для чтения из последовательного порта.
- Для закрытия последовательного порта нужно использовать функцию close().
- Конфигурирование последовательного порта предполагает изменение определенных атрибутов.

ПРИМЕРЫ

- Большинство систем поддерживает последовательный интерфейс POSIX для изменения параметров, таких как скорость передачи, размер символов и т.д.
- Две наиболее важные функции POSIX: **`tcgetattr()`** и **`tcsetattr()`**. Они, соответственно, позволяют получить и установить терминальные атрибуты. Необходимо предоставить указатель на структуру **`termios`**, которая содержит все доступные опции для последовательного интерфейса.

ПРИМЕРЫ

Поля структуры termios

c_cflag	Управляющие опции
c_lflag	Опции линии
c_iflag	Опции ввода
c_oflag	Опции вывода
c_cc	Управляющие символы
c_ispeed	Скорость ввода в бодах
c_ospeed	Скорость вывода в бодах

ПРИМЕРЫ

- Поле c_cflag управляет скоростью передачи данных, количеством битов данных, четностью, стоп-битами, и установкой аппаратного управления потоком передачи данных. c_cflag содержит две опции, которые всегда должны быть установлены: CLOCAL и CREAD. Функции **`cfsetospeed()`** и **`cfsetispeed()`** предназначены для установки скорости передачи данных в структуре **`termios`**. Функция **`tcgetattr()`** заполняет структуру **`termios`** текущими установками конфигурации последовательного порта. После установки нужной конфигурации используется **`tcsetattr()`**. Константа TCSANOW указывает, что все изменения вступают в силу немедленно. Также существуют другие константы для ожидания завершения операций ввода и вывода (TCSADRAIN), или для очистки буферов ввода и вывода (TCSAFLUSH).

ПРИМЕРЫ

- Размер символов задается битами:
- options.c_cflag &= ~CSIZE; /* Маскирование битов размера символов */
- options.c_cflag |= CS8; /* Установка 8 битов данных */
- Поле локального режима c_lflag управляет, как вводимые символы будут обрабатываться драйвером последовательного порта. В основном приходится конфигурировать поле c_lflag для канонического или неканонического (raw) ввода.

ПРИМЕРЫ

Константы для поля c_lflag

ISIG	Разрешить SIGINTR, SIGSUSP, SIGDSUSP, и SIGQUIT сигналы
ICANON	Разрешить канонический ввод (иначе неканонический)
ECHO	Разрешить эхо вводимых символов
ECHOE	Символ эхо стирания как BS-SP-BS
NOFLSH	Блокировка очистки приемных буферов после символа прерывания или выхода

ПРИМЕРЫ

- **Выбор канонического ввода:**
- Канонический ввод - строчно-ориентированный. Вводимые символы помещаются в буфер который может интерактивно редактироваться пользователем до приема символа CR (carriage return) или LF (line feed). При выборе этого режима нормально устанавливаете опции ICANON, ECHO и ECHOE: options.c_lflag |= (ICANON | ECHO | ECHOE);
- **Выбор неканонического (Raw) ввода:**
- Неканонический ввод не обрабатывается. Вводимый символ передается без изменений, так как он был принят. В основном сбрасываете опции ICANON, ECHO, ECHOE и ISIG приустановке неканонического ввода:
- options.c_lflag &= ~(ICANON | ECHO | ECHOE | ISIG);

ПРИМЕРЫ

- **Опции ввода:**
- Поле опций ввода c_iflag управляет любой обработкой ввода которая выполняется над символами принимаемыми из последовательного порта.
- *Константы для поля c_iflag*

INPCK	Производить проверку на четность
IGNPAR	Игнорировать проверка на четность
IGNCR	Игнорировать знак возврата каретки

ПРИМЕРЫ

- **Опции вывода:**
- Поле c_oflag содержит опции фильтрации вывода. Так же как для режима ввода, можно выбрать обработанный или не обработанный (raw) вывод данных. Обработанный вывод выбирается установкой опции OPOST в поле c_oflag: options.c_oflag |= OPOST;
- Выбор необработанного (raw) вывода устанавливается сбросом опции OPOST в поле c_oflag: options.c_oflag &= ~OPOST;
- Когда опция OPOST сброшена, все остальные биты опций в поле c_oflag игнорируются.

ПРИМЕРЫ

- **Управляющие символы:**
- Символьный массив c_cc содержит описания управляющих символов и параметры таймаутов. Константы объявлены для каждого элемента этого массива: VINTR – прерывание, VQUIT – выход, VMIN - минимальное количество символов для чтения, VTIME - время ожидания данных (десятые доли секунды)
- **Модем.**
- Модемы - это устройства которые модулируют последовательные данные в частоты, которые могут быть переданы по аналоговым линиям подобным телефонным линиям или кабелям TV подключений.

ПРИМЕРЫ

- **Управление модемом**
- Первый шаг в организации связи через модем - это открытие и конфигурирование порта для неструктурированного (raw) ввода, следующее что необходимо сделать, это установить связь с модемом. Наилучший способ сделать это посылкой модему "AT" команды. Когда модем корректно подключен и включен, он будет выдавать ответ "OK".
- **Стандартные команды модема**
- Большинство модемов использует множество "AT" команд, называемых так поскольку каждая команда начинается с символов "AT" и сопровождается специфическими командами и символом возврата каретки.

ПРИМЕРЫ

- После обработки команды модем будет отвечать одним из нескольких текстовых сообщений, в зависимости от команды.
- ATD - Набор номера (Dial A Number)
- Команда ATD производит набор указанного номера. В дополнение к цифрам и дефисам можете указать тип набора тональный ("T") или импульсный ("P"), паузу на одну секунду (","), и ожидание тональной посылки ("W"):
- Пример:
- ATDT 18008008008W1234,1,1234

ПРИМЕРЫ

- Модем ответит одним из следующих сообщений:
- NO DIALTONE
- BUSY
- NO CARRIER
- CONNECT
- CONNECT baud
- Из других команд модема отметим следующие:
- ATH - Повесить трубку (Hang Up)
- ATZ - Сброс модема.

ПРИМЕРЫ

- Методические указания:
- 1) Для функции ioctl необходимо подключить библиотеку sys/ioctl.h , для работы с последовательным портом библиотеку termios.h, для получения кодов ощибок - errno.h, для управления файлами - fcntl.h
- 2) Для инициализации порта рекомендуется написать отдельную функцию и потом вызывать ее перед передачей команд модему.
- 3) Перехватить ответ модема можно с помощью функций для работы со строками.
- 4) Необходимо отключить эхо ввода. Эхо ввода приведет к циклу обратной связи между модемом и компьютером.

ПРИМЕРЫ

- 5) При отправлении команды модему, необходимо завершать их символом возврата каретки (CR), а не символом новой строки (NL). Символьная константа - "\r".
- 6) Нужно убедиться, что используется скорость передачи, которую поддерживает модем. Хотя многие модемы выполняют автоматическое определение скорости передачи, некоторые имеют ограничения (общее ограничение 19.2kbps).
- 7) Засечь время выполнения программы можно любым из описанных выше способов.
- Код программы приведен здесь.

ПРИМЕРЫ

- **Опрос сетевой карты на шине PCI**
- Задание: опросить сетевую карту на шине PCI, получить ее mac-адрес.
- Вывести на экран время выполнения программы.
- Краткие теоретические сведения:
- Разработка шины PCI (Peripheral Component Interconnect bus) - шины для подсоединения периферийных устройств - началась в 1991 году как внутренний проект корпорации Intel. Разработчики Intel отказались от использования шины процессора и ввели еще одну "антресольную" (mezzanine) шину. Благодаря такому решению шина получилась процессоро-независимой, могла работать параллельно с шиной процессора, не обращаясь к ней за запросами, и, тем самым, снижая ее загрузку.

ПРИМЕРЫ

- Стандарт шины был объявлен открытым и передан PCI Special Interest Group (www.pcisig.com), которая продолжила работу по совершенствованию шины.
- Основные возможности шины следующие:
- - синхронный 32-х или 64-х разрядный обмен данными. При этом для уменьшения числа контактов (и стоимости) используется мультиплексирование, то есть адрес и данные передаются по одним и тем же линиям.
- - поддержка 5V и 3.3V логики. Частота 66MHz поддерживается только 3.3V логикой.
- - частота работы шины 33MHz или 66MHz позволяет обеспечить широкий диапазон пропускных способностей (с использованием пакетного режима):

ПРИМЕРЫ

- 132 MB/сек при 32-bit/33MHz;
- 264 MB/сек при 32-bit/66MHz;
- 264 MB/сек при 64-bit/33MHz;
- 528 MB/сек при 64-bit/66MHz.
- При этом для работы шины на частоте 66MHz необходимо, чтобы все периферийные устройства работали на этой частоте.
- - полная поддержка multiply bus master (например, несколько контроллеров жестких дисков могут одновременно работать на шине).
- - автоматическое конфигурирование карт расширения при включении питания.

ПРИМЕРЫ

- - спецификация шины позволяет комбинировать до восьми функций на одной карте (например, видео + звук и т.д.).
- - шина позволяет устанавливать до 4 слотов расширения, однако возможно использование моста PCI-PCI для увеличения количества карт расширения.
- - PCI-устройства оборудованы таймером, который используется для определения максимального промежутка времени, в течении которого устройство может занимать шину.
- Шина поддерживает метод передачи данных, называемый "linear burst" (метод линейных пакетов). Этот метод предполагает, что адрес автоматически увеличивается для следующего байта, при этом увеличивается скорость передачи данных за счет уменьшения числа передаваемых адресов.

ПРИМЕРЫ

- Согласно спецификации, каждое устройство PCI имеет конфигурационное пространство размером 256 байт, в котором содержится информация о самом устройстве и о ресурсах, занимаемых устройством.
- Получим мак-адрес сетевой платы, используя сокеты.
- Далее с помощью директивы функции ioctl SIOCGIFHWADDR запишем мак-адрес в заранее определенную структуру.
- Методические указания
- 1) Для работы с сетевой картой рекомендуется использовать сокеты, функции работы с памятью.
- 2) Для функции ioctl необходимо подключить библиотеку sys/ioctl.h , для работы с сокетами - библиотеку linux/if.h и sys/socket.h

ПРИМЕРЫ

- 3) Описание функций socket(), memset() и др. можно посмотреть с помощью команды man.
- 4) Локальный интерфейс, который через общую память эмулирует работу сетевой карты - eth0.
- 5) Засечь время выполнения программы можно любым из описанных выше способов.
- Код программы приведен здесь.

ПРИМЕРЫ

- **Опрос и управление CD-ROM**
- Задание: опросить устройство cd-rom, получить информацию о проигрываемом в данный момент файле (номер трека, время проигрывания), выдвинуть и задвинуть лоток cd-rom. Вывести на экран время выполнения программы.
- Краткие теоретические сведения:
- Изначально лазерные компакт-диски разрабатывались именно как носители оцифрованного звука. Практически любое устройство чтения CD-ROM является по совместительству и плеером аудио CD и позволяет воспроизводить аудиодиски при минимальном вмешательстве со стороны системы.

ПРИМЕРЫ

- Драйверы устройств чтения CD-ROM предоставляют как функции контроля воспроизведения аудиодисков средствами, так и функции непосредственного чтения аудиоданных (эти функции используются в основном программами-рипперами (rippers)). В Linux интерфейс драйвера CD-ROM описан в файле linux/cdrom.h.
- **Воспроизведение аудио CD.**
- Запись на любом компакт-диске состоит из нескольких треков. Треки нумеруются начиная с нуля (трек 0 содержит оглавление диска). Номер трека не может превышать значение 99. На аудио CD каждый музыкальный фрагмент как правило записывается на отдельном треке.

ПРИМЕРЫ

- В определенных ситуациях одно произведение может быть записано на нескольких треках, или же наоборот, на одном треке может быть записано несколько независимых фрагментов. Последний вариант применяется, когда число фрагментов, которые необходимо записать на диск, превышает 99. В этом случае, для различения фрагментов внутри одного трека используются индексы. На одном и том же диске могут быть записаны как аудио-данные, так и другая информация. Перед воспроизведением трека с такого "смешанного" диска следует проверять, является ли трек аудио-треком или треком данных.

ПРИМЕРЫ

- Запись на диске разбивается на фреймы. Каждый фрейм содержит 2352 байта. Для обеспечения указанных стандартных характеристик цифровой записи чтение данных должно выполняться со скоростью 75 фреймов в секунду (что соответствует однократной скорости чтения CD-ROM). С фреймами связан один из форматов адресации на аудио CD. Адресация осуществляется в единицах MSF - минуты, секунды, фреймы - где фрейм можно рассматривать как 1/75 секунды.
- В нижеследующей таблице приводятся основные вызовы ioctl, связанные с воспроизведением аудио CD.

ПРИМЕРЫ

- *Директивы ioctl для CD-rom*

Вызов	Описание	Доп. параметр
CDROM_DRIVE_STATUS	Получение данных о состоянии устройства	константа CDSL_XXX
CDROM_DISC_STATUS	Получение данных о диске	константа CDSL_XXX
CDROMREADTOCHDR	Чтение заголовка оглавления диска	структура cdrom_tochdr

ПРИМЕРЫ

Вызов	Описание	Доп. параметр
CDROMREADTOCENTRY	Чтение элемента оглавления диска	структура cdrom_tocentry
CDROMSUBCHNL	Чтение данных о параметрах воспроизведения	структура cdrom_subchnl
CDROMPLAYTRKIND, CDROMPLAYMSF	Воспроизведение аудиозаписи	Структуры cdrom_ti и cdrom_msf

ПРИМЕРЫ

Вызов	Описание	Доп. параметр
CDROMSTOP	Остановка воспроизведения	значение 0
CDROMPAUSE, CDROMRESUME	Приостановка, возобновление воспроизведения	значение 0
CDROMEJECT	Открытие лотка устройства	значение 0
CDROMCLOSETRAY	Закрытие лотка устройства	значение 0

ПРИМЕРЫ

- Вызовы CDROM_DRIVE_STATUS и CDROM_DISC_STATUS отличаются тем, что результат возвращается не в параметре-ссылке, а как значение функции ioctl. В качестве третьего аргумента ioctl выступает одна из констант, определенных в файле cdrom.h. Эти константы предназначены для работы с устройствами автоматической смены компакт-дисков (CD changers).
- В случае "однодискового" устройства следует использовать CDSL_CURRENT. Результатом вызова CDROM_DRIVE_STATUS могут быть значения CDS_NO_DISC (нет диска в устройстве), CDS_DRIVE_NOT_READY (устройство не готово), CDS_DISC_OK (диск обнаружен), а также некоторые другие константы из файла cdrom.h.

ПРИМЕРЫ

- Среди значений, возвращаемых вызовом CDROM_DISC_STATUS следует отметить CDS_NO_DISC (нет диска в устройстве) CDS_AUDIO (диск опознан как аудио) и CDS_MIXED (диск опознан как "смешанный"). Остальные значения соответствуют не-аудиодискам.
- Вызовы CDROMREADTOCHDR и CDROMREADTOCENTRY предназначены для работы с оглавлением диска. Вызов CDROMREADTOCHDR позволяет получить данные о номере первого и последнего информационных треков на диске, а вызов CDROMREADTOCENTRY - данные об отдельном треке - адрес начала трека (в формате MSF или LBA), тип трека (аудио или данные) и т.п.

ПРИМЕРЫ

- Вызов CDROMSUBCHNL позволяет получить информацию о текущем состоянии устройства - находится ли диск в режиме воспроизведения, и в какой позиции выполняется чтение данных.
- Вызовы CDROMPLAYTRKIND и CDROMPLAYMSF запускают воспроизведение аудиозаписи. Первый вызов позволяет задать начало и конец воспроизводимого фрагмента значениями трек/индекс, второй - адресами в формате MSF. Поскольку оглавление диска содержит данные о начальных адресах треков, воспроизведение отдельного трека часто выполняется по принципу "от начала данного трека до начала следующего". В файле cdrom.h определена константа CDROM_LEADOUT, указывающая на условный трек, расположенный после последнего трека.

ПРИМЕРЫ

- Вызовы CDROMSTOP, CDROMPAUSE и CDROMRESUME выполняют, соответственно, остановку, временную остановку (пауза) и возобновление воспроизведения.
- Методические указания
- 1) Для функции ioctl необходимо подключить библиотеку sys/ioctl.h , для работы с файлами заголовочный файл sys/fcntl.h для работы с CD-ROM - библиотеку linux/cdrom.h
- 2) Для получения информации используйте стандартные структуры, определенные в cdrom.h
- 3) Обратите внимание, что такие функции как открытие лотка CDrom в функции ioctl не имеют третьего параметра.
- Код программы приведен здесь.

ПРИМЕРЫ

- **Параллельный порт. Управление принтером.**
- Задание: опросить принтер через параллельный порт, вывести на печать произвольный текст, вывести на экран состояние системных регистров до и после печати
- Краткие теоретические сведения:
- Порт параллельного интерфейса был введен в PC для подключения принтера — LP'T-порт (Line PrinTer — построчный принтер).
- Адаптер параллельного интерфейса представляет собой набор регистров, расположенных в пространстве ввода/вывода. Регистры порта адресуются относительно базового адреса порта, стандартными значениями которого являются 386h, 378h и 278h.

ПРИМЕРЫ

- Порт имеет внешнюю 8-битную шину данных, 5-битную шину сигналов состояния и 4-битную шину управляющих сигналов. BIOS поддерживает до четырех LPT-портов своим сервисом — прерыванием INT 17h, обеспечивающим через них связь с принтерами по интерфейсу Centronics. Этим сервисом BIOS осуществляет вывод символа, инициализацию интерфейса и принтера, а также опрос состояния принтера.
- В Linux параллельным портам сопоставляются файлы /dev/lp0, /dev/lp1.
- Опрос регистров принтера и выдачу произвольной информации на печать можно произвести несколькими способами, первый – использовать функцию ioctl(), директивы для нее описаны в файле parport.h

ПРИМЕРЫ

- PPWDATA – запись информации в параллельный порт
- PPRCONTROL – контроль чтения из параллельного порта
  PPRSTATUS - статус последней команды чтения из параллельного порта
- Второй метод – непосредственная запись/чтение в порт и из порта с помощью следующих функций [9]:
- outb, outw, outl, outsb, outsw, outsl - функции вывода данных в порт
  inb, inw, inl, insb, insw, insl - функции чтения данных порта
  outb_p, outw_p, outl_p, inb_p, inw_p, inl_p - функции задержки ввода/вывода.

ПРИМЕРЫ

- Это семейство функций используется для низкоуровневой работы с портами ввода/вывода. В основном, они предназначены для использования внутри ядра, но могут быть вызваны и пользовательской программой. Для того, чтобы запросить у ядра разрешение доступа к портам, нужно использовать ioperm() или iopl(). Если этого не сделать, приложение получит сообщение об ошибке сегментации. Для печати можно использовать функцию write() : write(fd,"\nTest\r",sizeof("\nTest\r")), аргументами которой являются дескриптор устройства, сообщение и длина сообщения.

ПРИМЕРЫ

- Методические указания
- 1) Для функции ioctl необходимо подключить библиотеку sys/ioctl.h, для работы с параллельным портом linux/ppdev.h, linux/parport.h, для прямой записи / чтения – заголовочный файл sys/io.h
- 2) Описания функций inb(), outb(), iopl() можно посмотреть, используя команду man.
- 3) При отправлении текста на печать, необходимо завершать его символом возврата каретки. Символьная константа - "\r".
- 5) В конце записи необходимо отправить в порт команду окончания печати 0xFF
- 6) Засечь время выполнения программы можно любым из описанных выше способов.
- Код программы приведен здесь.