Министерство образования и науки РФ

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ

УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра ЭМИС

Касимов В.З.

СИСТЕМЫ РЕАЛЬНОГО ВРЕМЕНИ

Методические рекомендации по выполнению лабораторных и самостоятельных работ

2012

Касимов В.З.

Системы реального времени. Лабораторные и самостоятельные работы.

Методические рекомендации. – Томск: ТУСУР, 2012. – 11 с.

Методические рекомендации посвящены программированию в ОС UNIX. Пособие рассчитано на студентов вузов.

СОДЕРЖАНИЕ

Лабораторная работа №1 Знакомство с утилитами командной строки POSIX совместимой операционной системы реального времени.

Цель работы: Получить навыки работы с основными утилитами командной строки.

Время выполнения: 2ч.

Программное обеспечение: любая POSIX совместимой операционной системы (Linux, FreeBsd)

Лабораторная работа №2 Процессы в Unix-подобных операционных системах.

Цель работы: Познакомится с интерфейсом управления процессами системным вызовом fork(), и семейством системных вызовов exec();

Время выполнения: 6ч.

Программное обеспечение: любая POSIX совместимой операционной системы (Linux, FreeBsd)

Задание:

1. В качестве примера использования системных вызовов getpid() и getppid() самостоятельно напишите программу, печатающую значения PID и PPID для текущего процесса. Запустите ее несколько раз подряд. Посмотрите, как меняется идентификатор текущего процесса. Объясните наблюдаемые изменения.
2. Наберите программу (листиг 3.1.) откомпилируйте ее и запустите на исполнение (лучше всего это делать не из оболочки mc, так как она не очень корректно сбрасывает буферы ввода-вывода). Проанализируйте полученный результат.
3. Измените предыдущую программу с fork() так, чтобы родитель и ребенок совершали разные действия, например, выводили на экран разные строки.
4. Напишите программу, распечатывающую значения аргументов командной строки и параметров окружающей среды для текущего процесса.
5. Наберите, откомпилируйте программу (листиг 3.2.) и запустите на исполнение. Поскольку при нормальной работе будет распечатываться содержимое файла с именем 03-2.c, такой файл при запуске должен присутствовать в текущей директории (проще всего записать исходный текст программы под этим именем). Проанализируйте результат.
6. Для закрепления полученных знаний модифицируйте программу, созданную при выполнении задания раздела «Написание, компиляция и запуск программы с использованием вызова fork() с разным поведением процессов ребенка и родителя» так, чтобы порожденный процесс запускал на исполнение новую (любую) программу.

Вопросы для самоконтроля

1. В чем разница между процессом и программой?
2. Каковы алгоритмы запуска нового процесса и новой программы?
3. В чем заключается выгода от использования псевдопараллельности?

Лабораторная работа №3 Организация взаимодействия процессов через pipe и FIFO в UNIX.

Цель работы: Познакомится с интерфейсом управления процессами системным вызовом fork(), и семейством системных вызовов exec();

Время выполнения: 6ч.

Программное обеспечение: любая POSIX совместимой операционной системы (Linux, FreeBsd)

Задание:

1. Наберите, откомпилируйте программу (листинг 5.1) и запустите ее на исполнение. Обратите внимание на использование системного вызова umask() с параметром 0 для того, чтобы права доступа к созданному файлу точно соответствовали указанным в системном вызове open().
2. Измените программу из предыдущего пункта задания так, чтобы она читала записанную ранее в файл информацию и печатала ее на экране. Все лишние операторы желательно удалить.
3. Наберите программу иллюстрирующую работу с pip'ом в рамках одного процесса (листинг 5.2), откомпилируйте ее и запустите на исполнение.
4. Наберите, откомпилируйте и запустите на исполнение программу, осуществляющую однонаправленную связь через pipe между процессом-родителем и процессом-ребенком (листинг 5.3).
5. Модифицируйте программу(листинг 5.3) для связи между собой двух родственных процессов, исполняющих разные программы.
6. Определите размер pipe для вашей операционной системы.
7. Наберите, откомпилируйте и запустите на исполнение программу (листинг 5.4.), осуществляющую однонаправленную связь через FIFO между процессом-родителем и процессом-ребенком
8. Для закрепления полученных знаний напишите на базе предыдущего примера две программы, одна из которых пишет информацию в FIFO, а вторая – читает из него, так чтобы между ними не было ярко выраженных родственных связей (т.е. чтобы ни одна из них не была потомком другой).

Вопросы для самоконтроля

1. Раскройте понятие потока ввода-вывода.
2. Опишите алгоритмы работы с файлами через системные вызовы и стандартную библиотеку ввода-вывода.
3. Раскройте понятие файлового дескриптора.
4. Раскройте понятие pipe(Раскройте понятие FIFO).

5. Опишите алгоритм организация связи через pipe между процессом-родителем и процессом-потомком.
6. Опишите алгоритм организация связи через FIFO между процессом-родителем и процессом-потомком.
7. Отметьте особенности поведения вызовов read() и write() для pip'a и FIFO.
8. Расскажите про особенности поведения вызова open() при открытии FIFO.

Лабораторная работа №4 Средства System V IPC. Организация работы с разделяемой памятью в UNIX. Понятие нитей исполнения (thread)

Цель работы: Знакомство с организацией работы с разделяемой памятью.

Время выполнения: 4 ч.

Программное обеспечение: любая POSIX совместимой операционной системы (Linux, FreeBsd)

Задание:

1. Наберите программы (листинг 6.1a, 6.1b) для иллюстрирующие работы с разделяемой памятью, сохраните под именами 06-1a.c и 06-1b.c соответственно, откомпилируйте их и запустите несколько раз. Проанализируйте полученные результаты.
2. Получите информацию с помощью команды ipcs о средствах System V IPC, к которым вы имеете право доступа на чтение. Удалите созданный вами сегмент разделяемой памяти из операционной системы, используя команды ipcrm
3. Осуществите сборку исполняемого файла программы (листинг 6.2.) иллюстрирующей работу двух нитей. Для сборки исполняемого файла при работе редактора связей необходимо явно подключить библиотеку функций для работы с pthread'ами, которая не подключается автоматически. Это делается с помощью добавления к команде компиляции и редактирования связей параметра -lpthread – подключить библиотеку pthread. Наберите текст, откомпилируйте эту программу и запустите на исполнение.
4. Модифицируйте предыдущую программу, добавив к ней третью нить исполнения.

Вопросы для самоконтроля

1. Перечисленные преимущества и недостатки потокового обмена данными.
2. Раскройте понятие System V IPC.
3. Опишите алгоритм механизма разделяемой памяти
4. Раскройте понятие о нити исполнения (thread) в UNIX.
5. Опишите алгоритм использования нитей.

Лабораторная работа №5 Организация файловой системы в UNIX. Работа с файлами и директориями. Понятие о memory mapped файлах

Цель работы: Практическое применение команд и системных вызовов для операций над файлами

Время выполнения: 4 ч.

Программное обеспечение: любая POSIX совместимой операционной системы (Linux, FreeBsd)

Задания:

1. Напишите, откомпилируйте и прогоните программу, распечатывающую список файлов, входящих в директорию, с указанием их типов. Имя директории задается как параметр командной строки. Если оно отсутствует, то выбирается текущая директория.
2. Напишите программу, распечатывающую содержимое заданной директории в формате, аналогичном формату выдачи команды ls -al. Для этого вам дополнительно понадобится самостоятельно изучить в UNIX Manual функцию ctime(3) и системные вызовы time(2), readlink(2). Цифры после имен функций и системных вызовов – это номера соответствующих разделов для UNIX Manual.
3. Наберите программу (листинг 11.1) для иллюстрации работы с memory mapped файлом, откомпилируйте и запустите. Проанализируйте полученные результаты.
4. Модифицируйте программу из предыдущего задания так, чтобы она отображала файл, записанный программой из раздела "Анализ, компиляция и прогон программы для создания memory mapped файла и записи его содержимого", в память и считала сумму квадратов чисел от 1 до 100000, которые уже находятся в этом файле.
5. Напишите две программы, использующие memory mapped файл для обмена информацией при одновременной работе, подобно тому, как они могли бы использовать разделяемую память

Вопросы для самоконтроля

1. Перечислите основные типы файлов в Unix.
2. В чем отличие файлов отображаемых в память от механизма разделяемой памяти?
3. В чем отличие файлов отображаемых в память от потоковых средств связи, таких как регулярные файлы и FIFO?
4. Перечислите специальные функции для работы с содержимым директорий.

Лабораторная работа №6 Семейство протоколов TCP/IP. Сокеты (sockets) в UNIX и основы работы с ними

Цель работы: Организация связи между удаленными процессами с помощью датаграмм и установки логического соединения.

Время выполнения: 4 ч.

Программное обеспечение: любая POSIX совместимой операционной системы (Linux, FreeBsd)

Задания:

1. Наберите и откомпилируйте программу (Листинг 15-16.1). простого UDP клиента для сервиса echo. Перед запуском **"узнайте у своего системного администратора"**, запущен ли в системе стандартный UDP-сервис echo и если нет, попросите стартовать его. Запустите программу с запросом к сервису своего компьютера, к сервисам других компьютеров. Что произойдет, если в качестве IP-адреса указать несуществующий адрес, адрес выключенной машины или машины, на которой не работает сервис echo.

2. Наберите и откомпилируйте программу (Листинг 15-16.2). простого UDP сервера для сервиса echo. Запустите ее на выполнение. Модифицируйте текст программы UDP-клиента , заменив номер порта с 7 на 51000. Запустите клиента с другого виртуального терминала или с другого компьютера и убедитесь, что клиент и сервер взаимодействуют корректно.
3. Наберите и откомпилируйте программу (Листинг 15-16.3) . простого TCP-клиента для сервиса echo. Перед запуском выясните, запущен ли в системе стандартный TCP-сервис echo и, если нет, запустите его. Запустите программу с запросом к сервису своего компьютера, к сервисам других компьютеров. Каков будет результат выполнения программы, если в качестве IP-адреса указать несуществующий адрес или адрес выключенной машины?

4. Наберите и откомпилируйте программу(Листинг 15-16.4). простого TCP сервера для сервиса echo... Запустите ее на выполнение. Модифицируйте текст программы TCP-клиента (листинг 15-16.3), заменив номер порта с 7 на 51000. Запустите клиента с другого виртуального терминала или с другого компьютера и убедитесь, что клиент и сервер взаимодействуют корректно.
5. Напишите, откомпилируйте и запустите параллельный сервер схема работы которого приведена на рисунке 15-16.9. Убедитесь в его работоспособности. Не забудьте о необходимости удаления зомби-процессов.
6. Наберите программы(Листинг15–16-6, 15–16-5), откомпилируйте, убедитесь в работоспособности и проанализируйте результаты.
7. По аналогии с программами в предыдущем примере модифицируйте тексты программ TCP клиента и сервера для сервиса echo (пример 15-16.3 и пример 15-16.4) для потокового общения в семействе UNIX Domain протоколов. Откомпилируйте их и убедитесь в правильном функционировании.

Вопросы для самоконтроля

1. Перечислите основные достоинства и недостатки установление соединения при помощи датаграмм и установки логического соединения.
2. Поясните понятие сокета. Что такое дескриптор сокета?
3. Опишите процедуру трехэтапного рукопожатия
4. Опишите алгоритм работы простейшего tcp –сервера с параллельной обработкой запросов клиентов

Лабораторная работа №7 Организация ввода-вывода в UNIX. Файлы устройств. Аппарат прерываний. Сигналы в UNIX

Цель работы: Познакомится с понятием сигналов в POSIX совместимой операционной системы

Время выполнения: 4 ч.

Программное обеспечение: любая POSIX совместимой операционной системы (Linux, FreeBsd)

Задания:

1. Наберите программу(листинг 13-14.1) иллюстрирующую понятия группа процессов, сеанс, фоновая группа, откомпилируйте ее и запустите на исполнение (лучше всего из-под оболочки Midnight Commander – mc). Запустив команду "ps –e j" с другого экрана, проанализируйте значения идентификаторов группы процессов, сеансов, прикрепления управляющего терминала, текущей и фоновой групп. Убедитесь, что тривиальные процессы относятся к текущей группе сеанса. Проверьте реакцию текущей группы на сигналы SIGINT – нажатие клавиш <CTRL> и <C> – и SIGQUIT – нажатие клавиш <CTRL> и <4>.
2. Запустите программу из предыдущего задания в фоновом режиме, например командой "a.out &". Проанализируйте значения идентификаторов группы процессов, сеансов, прикрепления управляющего терминала, текущей и фоновой групп. Убедитесь, что тривиальные процессы относятся к фоновой группе сеанса. Проверьте реакцию фоновой группы на сигналы SIGINT – нажатие клавиш <CTRL> и <C> – и SIGQUIT – нажатие клавиш <CTRL> и <4>. Ликвидируйте тривиальные процессы с помощью команды kill.
3. Запустите программу на исполнение из-под Midnight Commander в текущей группе. Проанализировав значения идентификаторов группы процессов, сеансов, прикрепления управляющего терминала, текущей и фоновой групп, ликвидируйте лидера сеанса для тривиальных процессов. Убедитесь, что все процессы в текущей группе этого сеанса прекратили свою работу.
4. Запустите программу в фоновом режиме. Снова удалите лидера сеанса для тривиальных процессов. Убедитесь, что фоновая группа продолжает работать. Ликвидируйте тривиальные процессы.
5. Наберите программу(листинг 13–14-2.c) Наберите, откомпилируйте и запустите эту программу, убедитесь, что на нажатие клавиш <CTRL> и <C> она не реагирует, а реакция на нажатие клавиш <CTRL> и <4> осталась прежней.
6. Модифицируйте программу из предыдущего раздела так, чтобы она перестала реагировать и на нажатие клавиш <CTRL> и <4>. Откомпилируйте и запустите ее, убедитесь в отсутствии ее реакций на внешние раздражители. Снимать программу придется теперь с другого терминала командой kill.
7. Наберите, откомпилируйте и запустите программу с пользовательской обработкой сигнала SIGINT (листинг 13–14-3), проверьте ее реакцию на нажатие клавиш <CTRL> и <C> и на нажатие клавиш <CTRL> и <4>.
8. Модифицируйте программу из предыдущего задания так, чтобы она печатала сообщение и о нажатии клавиш <CTRL> и <4>. Используйте одну и ту же функцию для обработки сигналов SIGINT и SIGQUIT. Откомпилируйте и запустите ее, проверьте корректность работы. Снимать программу также придется с другого терминала командой kill.
9. Наберите, откомпилируйте и запустите программу с пользовательской обработкой сигнала SIGINT (листинг 13–14-4)

10. Организуйте двустороннюю поочередную связь процесса-родителя и процесса-ребенка через pipe, используя для синхронизации сигналы SIGUSR1 и SIGUSR2, модифицировав программу из раздела. "Прогон программы для организации однонаправленной связи между родственными процессами через pipe" семинара 5.
11. Организуйте побитовую передачу целого числа между двумя процессами, используя для этого только сигналы SIGUSR1 и SIGUSR2
12. Для закрепления материала наберите программу (листинг 13–14-5),.c демонстрирующую асинхронное получением информации о статусе завершения порожденного процесса.
13. Откомпилируйте программу(листинг 13–14-6), прогоните и посчитайте количество сообщений о статусе завершившихся “детей”. Проанализируйте результат.
14. Модифицируйте обработку сигнала в программе из предыдущего задания, не применяя POSIX-сигналы, так, чтобы процесс-родитель все-таки сообщал о статусе всех завершившихся процессов-детей.

Вопросы для самоконтроля

1. Поясните понятие виртуальной файловой системы.
2. . Поясните понятие драйвера. Перечислите типы драйверов.
3. Раскройте понятие сигнала. Перечислите основные способы возникновения сигналов и виды их обработки.
4. Понятия группы процессов, сеанса, лидера группы, лидера сеанса, управляющего терминала сеанса.
5. Поясните, каким образом сигналы используются для синхронизации процессов.