Лабораторный практикум

Выполняется при помощи уникальной системы мониторинга операционных систем **SAMOS**.

Система позволяет в реальном масштабе времени анализировать основные структуры данных ядра при выполнении основных системных вызовов по управлению процессами, файлами, сигналами и каналами.

Домашняя подготовка

Иметь в распечатанном виде описание системы **SAMOS** и описание соответствующей лабораторной работы.

Лабораторные работы можно выполнять дистанционно по **ssh** или **putty**.

Сервер: **samos.dozen.mephi.ru**.

Порт: **22**.

Протокол: **ssh**.

Лабораторная работа № 1 Управление файлами

Создание, открытие и закрытие файлов (1)

- Запустите систему поддержки и создайте UNIX-процесс (команда меню **FileRNew** в **Process Manager**). Появится новое окно, соответствующее реально существующему в системе процессу. В этой лабораторной работе все действия будут выполняться именно через это окно, создание дополнительных окон процессов не требуется. Созданное окно для удобства можно растянуть по вертикали и горизонтали.
- Добейтесь, чтобы в текущей директории не было ни одного созданного заранее файла (меню **ShellRls -l** и **ShellRrm**).

Создание, открытие и закрытие файлов (2)

- С помощью системного вызова **creat** создайте файл **f1** с правами доступа **777**.
- Определите вектор характеристик созданного файла **f1** (дескриптор файла, режим открытия, указатель чтения/записи, количество ссылок на файл, временные характеристики, права доступа, индексный дескриптор). Воспользуйтесь набором пунктов меню **File** и системным вызовом **stat** (**SysCallRStat**).
- Получите ряд уже полученных характеристик другим способом – с помощью команд **Shell ls -l** и **ls -i**. Все ли характеристики совпали с уже полученными?

Создание, открытие и закрытие файлов (3)

- Обратите внимание на установленные в результате системного вызова **creat** права доступа. Совпадают ли они с затребованными? Если не совпадают, объясните почему и добейтесь совпадения. Используйте пункты меню **Shell** (**ViewUmask**, **Chmod**, другие по необходимости).
- Выполните еще один системный вызов **creat** и попытайтесь еще раз создать им файл **f1** с правами доступа **777**. Определите изменения, произошедшие в системе. Изменился ли вектор характеристик файла? Что именно изменилось?

Создание, открытие и закрытие файлов (4)

- Выполните еще один системный вызов **creat** и попытайтесь еще раз создать им файл f1 с правами доступа **666**. Определите изменения, произошедшие в системе. Изменился ли вектор характеристик файла? Что именно изменилось?
- Создайте файл **f2** с помощью системного вызова **open** (добейтесь создания нового файла, используя различные флаги). Определите вектор характеристик файла. Есть ли разница в создании файла с помощью **creat** и с помощью **open**? Если есть, то какая?

Создание, открытие и закрытие файлов (5)

- Закройте все созданные и открытые файлы, кроме открытых по умолчанию (то есть дескрипторы стандартного ввода, вывода и ошибки; перед тем, как начать закрывать файлы, убедитесь, что вы точно знаете, какие дескрипторы соответствуют стандартному вводу, выводу и ошибке).

Операции чтения и записи с файлами (1)

- С помощью creat создайте файл **f3** с правами доступа **000**. Определить его атрибуты по вышеприведенной схеме.
- Попытайтесь записать в этот файл (с начала файла) некоторый текст длиной 12 символов.
- Примечание. Для более устойчивой работы системы поддержки для операций записи использовать только буквы латинского алфавита и цифры. Не пытайтесь записывать более 100 символов одновременно.

Операции чтения и записи с файлами (2)

- Проанализируйте реакцию системы (атрибуты файла, его содержимое) и объясните ее.
- Попытайтесь прочитать 5 символов с начала файла.
- Попытайтесь открыть файл **f3** с флагом "только на чтение". Проанализируйте и объясните реакцию системы.

Операции чтения и записи с файлами (3)

- Примечание. Если здесь и далее система в ответ на какой-либо запрос выдает **errno** отличный от нуля, можно попробовать получить короткую расшифровку ошибки с помощью функции **Strerror** (см. Меню **Error**). Следует иметь в виду, что выдается лишь стандартная строка описания ошибки. Это описание не всегда может дать корректное объяснение текущей ситуации, хотя почти всегда дает намек. Текст сообщения тот же самый, который можно получить с помощью функции **perror**.

Операции чтения и записи с файлами (4)

- Добейтесь открытия файла **f3** с флагом "только на чтение" (возможно, придется изменить некоторые атрибуты файла).
- Попытайтесь еще раз выполнить предыдущие пункты. Попытайтесь записать в конец файла **f3** 10 символов. Объясните реакцию системы.
- Добейтесь выполнения предыдущего пункта (возможно придется изменить какие-то характеристики файла **f3**). Попытайтесь прочитать эти 10 символов с помощью системного вызова **read**. Просмотрите текущее содержимое файла с помощью команды **shell cat**.

Операции чтения и записи с файлами (5)

- Попытайтесь записать в файл **f3** 15 произвольных символов, которые должны следовать после дыры в 8192 символа. Просмотрите содержимое файла с помощью команды **cat**.
- Посмотрите распределение дисковой памяти (**FilesRInode**).
- Примечание. Возможна некоторая рассинхронизация показа информации с помощью (**FilesRInode**). По текущим оценкам не более одной минуты.
- Выполните предыдущие пункты, но дыра должна быть 8192*12=98304.

Операции чтения и записи с файлами (6)

- Зафиксируйте текущие атрибуты файла **f3** и далее сдублируйте дескриптор файла **f3**. Зафиксируйте возможные изменения в системе.
- Попытайтесь выполнить операцию чтения (системный вызов **read**) через основной и дубль-дескриптор. Сделайте выводы о возможностях системного вызова **dup**.
- Убедитесь, что файл **f3** имеет ненулевую длину. С помощью **creat** попытайтесь создать файл **f3** с правами доступа, равными текущим. Что изменилось в системе?

Операции чтения и записи с файлами (7)

- В результате наблюдения за особенностями работы **creat** сделайте вывод о его поведении в различных ситуациях. Предложите форму вызова **open** со всеми необходимыми флагами, которая была бы полностью эквивалентна **creat**. Проверьте эту форму с файлом **f4**.
- Закройте все файловые дескрипторы, кроме стандартного ввода, вывода и ошибки.

Исследование временных параметров файлов (1)

- Откройте файл **f3** на чтение/запись. Зафиксируйте временные характеристики файла.
- Посмотрите атрибуты файла **f3** с помощью команды **shell ls -l**. Вновь зафиксируйте временные характеристики файла.
- Запишите в начало файла **f3** произвольный текст в 10 символов. Зафиксируйте временные характеристики файла.

Исследование временных параметров файлов (2)

- Прочитайте с начала файла **f3** 5 символов. Зафиксируйте временные характеристики файла.
- Смените права доступа у файла **f3** на **766**. Зафиксируйте временные характеристики файла.

Исследование временных параметров файлов (3)

- Выполните команду **touch** в отношении файла **f3** и нового файла **f5**. Зафиксируйте временные характеристики файлов **f3** и **f5**, а также их длины. Сделайте вывод о возможностях команды **touch**.
- Закройте все файловые дескрипторы, кроме стандартного ввода, вывода и ошибки.

Удаление файла (1)

- Убедитесь в наличии файла **f3** в текущем каталоге.
- Дважды откройте файл **f3** на чтение и на запись соответственно.
- Установите указатель чтения/записи на начало файла **f3** и уничтожьте файл **f3** по команде **rm**.
- Проверьте изменения в системе относительно файла **f3** после выполнения команды **rm**. Примечание. Обратите внимание на значение счетчика ссылок на файл.

Удаление файла (2)

- Убедитесь, что файл **f3** отсутствует в текущем каталоге.
- Проверьте возможность доступа к уничтоженному файлу через его дескрипторы: попытайтесь записать с начала файла **f3** и прочитать из него 12 символов.
- Закройте файл **f3** по возможности чтения и определите изменения в составе вектора характеристик этого файла.
- Проверьте возможность чтения и записи в отношении файла **f3**.
- Проверьте при этом наличие файла в текущем каталоге.

Удаление файла (3)

- Закройте файл 3 по возможности записи. Определите изменения в составе вектора характеристик этого файла. Сделайте вывод о возможностях операций закрытия и уничтожения файлов.

Лабораторная работа № 2 Управление процессами

Управление одним процессом (1)

- Создать исходный процесс.
- Определить вектор характеристик нового процесса:

  - ✔ Идентификаторы процесса, родительского процесса и идентификатор группы.
  - ✔ Состояние и флаги процесса
  - ✔ Параметры планирования процесса (используя опции меню **Process → Info, Shell → ps -l**)
  - ✔ Время запуска процесса
  - ✔ Диспозицию сигналов
  - ✔ Таблицу открытых файлов

Управление одним процессом (2)

- Исследовать возможность изменения поправки к приоритету. Установить диапазон изменения значений поправок к приоритету и определить максимальное значение приоритета.
- Завершить развитие текущего процесса.

Управление двумя параллельными процессами (1)

- Создать исходный процесс.
- Создать из исходного процесса параллельный дочерний процесс по схеме **fork**.
- Определить состав наследуемых характеристик для дочернего процесса. Проанализировать возможность наследования дочерним процессом от порождающего процесса значений текущего приоритета и поправки к приоритету.
- Проверить, каким образом влияет изменение текущего приоритета в отцовском процессе на текущий приоритет дочернего процесса. И наоборот.

Управление двумя параллельными процессами (2)

- Проверить текущее состояние процессов в системе и завершить исполнение дочернего процесса. Проверить изменения в состояниях процессов.
- Выполнить в контексте родительского процесса системный вызов **wait** и проверить какие изменения произошли в состояниях процессов.
- Создать из исходного процесса параллельный дочерний процесс по схеме **fork**.
- Выполнить в контексте родительского процесса системный вызов **wait**.

Управление двумя параллельными процессами (3)

- Проверить текущее состояние процессов в системе и завершить исполнение дочернего процесса . Проверить изменения в состояниях процессов.
- Создать из исходного процесса параллельный дочерний процесс по схеме **fork** → **exec**.
- Выполнить в контексте родительского процесса системный вызов **wait**.
- Проверить текущее состояние процессов в системе и завершить исполнение дочернего процесса. Проверить изменения в состояниях процессов.

Управление двумя параллельными процессами (4)

- Создать из исходного процесса параллельный дочерний процесс по схеме **fork**.
- Завершить исполнение родительского процесса и проверить текущее состояние процессов в системе.
- Завершить исполнение дочернего процесса.

Управление процессами и файлами (1)

- Создать исходный процесс и определить для него дескрипторы файлов, открытых по умолчанию.
- Создать из контекста текущего процесса новый файл.
- Закрыть файл с дескриптором 2.
- Открыть новый файл (на запись и чтение) и определить значение дескриптора, назначенного ему при открытии.
- Закрыть файл с дескриптором 1 и создать неименованный канал.

Управление процессами и файлами (2)

- Определить вектор открытых файлов для текущего процесса и определить: какие дескрипторы были назначены созданному неименованному каналу.
- Закрыть файл с дескриптором 0 и определить вектор открытых файлов для текущего процесса.
- Создать из исходного процесса параллельный дочерний процесс по схеме **fork** и определить доступный для него вектор открытых файлов.

Управление процессами и файлами (3)

- Проверить, каким образом влияет открытие и закрытие доступных файлов в отцовском процессе на вектор открытых файлов дочернего процесса. И наоборот.
- Завершить родительский и дочерний процессы.

Использование механизма сигналов (1)

- Создать исходный процесс.
- Изменить в контексте исходного процесса реакцию на сигналы:
  - ✔ сигнал 1 – игнорировать,
  - ✔ сигнал 2 – перехват и обработка с помощью предопределенной функции,
  - ✔ сигнал 9 – игнорировать.
- Выполнить посылку сигналов 1 и 2 из текущего процесса самому себе и проанализировать реакцию.
- Выполнить посылку сигнала 9 из текущего процесса самому себе и проанализировать реакцию.

Использование механизма сигналов (2)

- Создать исходный процесс.
- Изменить в контексте исходного процесса реакцию на сигналы:
  - ✔ сигнал 1 – игнорировать
  - ✔ сигнал 2 – перехват и обработка с помощью предопределенной функции.
- Создать параллельный дочерний процесс по схеме **fork** и определить наследуемую диспозицию сигналов.
- Проверить каким образом влияет изменение диспозиции сигналов в отцовском процессе на диспозицию сигналов в дочернем процессе. И наоборот.

Использование механизма сигналов (3)

- Из контекста родительского процесса послать сигнал 15 и определить текущее состояние процессов в системе.
- Выполнить в контексте родительского процесса системный вызов **wait** и определить текущее состояние процессов в системе.
- Создать параллельный дочерний процесс по схеме **fork** → **exec** и определить наследуемую диспозицию сигналов.