Операционные системы

Лабароторная работа №13

Гульдяев Тихон Дмитриевич

Содержание

Список таблиц

Список иллюстраций

1. Цель работы

Изучить основы программирования в оболочке ОС UNIX. Научиться писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

2. Выполнение лабораторной работы

Первая программа:

Написать командный файл, реализующий упрощённый механизм семафоров. Командный файл должен в течение некоторого времени t1 дожидаться освобождения ресурса, выдавая об этом сообщение, а дождавшись его освобождения, использовать его в течение некоторого времени t2<>t1, также выдавая информацию о том, что ресурс используется соответствующим командным файлом (процессом). Запустить командный файл в одном виртуальном терминале в фоновом режиме, перенаправив его вывод в другой (> /dev/tty#, где # — номер терминала куда перенаправляется вывод), в котором также запущен этот файл, но не фоновом, а в привилегированном режиме. Доработать программу так, чтобы имелась возможность взаимодействия трёх и более процессов

Код первой программы. (рис. 2.1).

```
$ sem.sh
#!/bin/bash

SEM_FILE="/tmp/sem.lock"

t1=12

t2=10

if [ ! -f "$SEM_FILE" ]; then
    echo "Ресурс занят. Ожидание освобождения ресурса..."
    sleep "$t1"
    while [ -f "$SEM_FILE" ]; do
        sleep 1
    done
fi

touch "$SEM_FILE"

echo "Ресурс доступен. Использование ресурса в течении $t2 секунд"
sleep "$t2"

rm "$SEM_FILE"

echo "Ресурс свободен"

```

Рис. 2.1: Код первой программы

В привелегированном режиме запущен на tty5, tty4, в фоновом в tty3(запускались в порядке tty5, tty4, tty3), перенаправление в следующий по счету. На рисунках отображены все используемые терминалы (рис. 2.2), (рис. 2.3), (рис. 2.4), (рис. 2.5).

```
guldyaev-tikhon@guldyaevtikhon:~/13$ ./sem.sh > /dev/tty4 &
[1] 3005
guldyaev-tikhon@guldyaevtikhon:~/13$
```

Рис. 2.2: Терминал tty3

```
guldyaev-tikhon@guldyaevtikhon:~/13$ sudo ./sem.sh > /dev/tty5
Ресурс занят. Ожидание освобождения ресурса...
guldyaev-tikhon@guldyaevtikhon:~/13$ Ресурс доступен. Использование ресурса в течении 10 секунд
Ресурс свободен
```

Рис. 2.3: Терминал tty4

```
guldyaev-tikhon@guldyaevtikhon:~/13$ sudo ./sem.sh > /dev/tty6
Ресурс занят. Ожидание освобождения ресурса...
guldyaev-tikhon@guldyaevtikhon:~/13$ Ресурс доступен. Использование ресурса в течении 10 секунд
Ресурс свободен
```

Рис. 2.4: Терминал tty5

```
guldyaev-tikhon@guldyaevtikhon:~$ Ресурс занят. Ожидание освобождения ресурса...
Ресурс доступен. Использование ресурса в течении 10 секунд
Ресурс свободен
```

Рис. 2.5: Терминал tty6

Вторая программа:

Реализовать команду man с помощью командного файла. Изучите содержимое каталога /usr/share/man/man1. В нем находятся архивы текстовых файлов, содержащих справку по большинству установленных в системе программ и команд. Каждый архив можно открыть командой less сразу же просмотрев содержимое справки. Командный файл должен получать в виде аргумента командной строки название команды и в виде результата выдавать справку об этой команде или сообщение об отсутствии справки, если соответствующего файла нет в каталоге man1.

Код второй программы (рис. 2.6)

$ man.sh

```
#!/bin/bash

if [ -z "$1" ]; then
    echo "Не указано название команды"
    exit 1
fi

command="$1"
man_dir="/usr/share/man/man1"
man_file="$man_dir/$command.1.gz"

if [ -f "$man_file" ]; then
    zcat "$man_file" | less
else
    echo "Справка для команды $command не найдена"
    exit 1
fi
```

Рис. 2.6: Код второй программы и скрипта

Пример использования второй программы, получение справки для ls. (рис. 2.7).

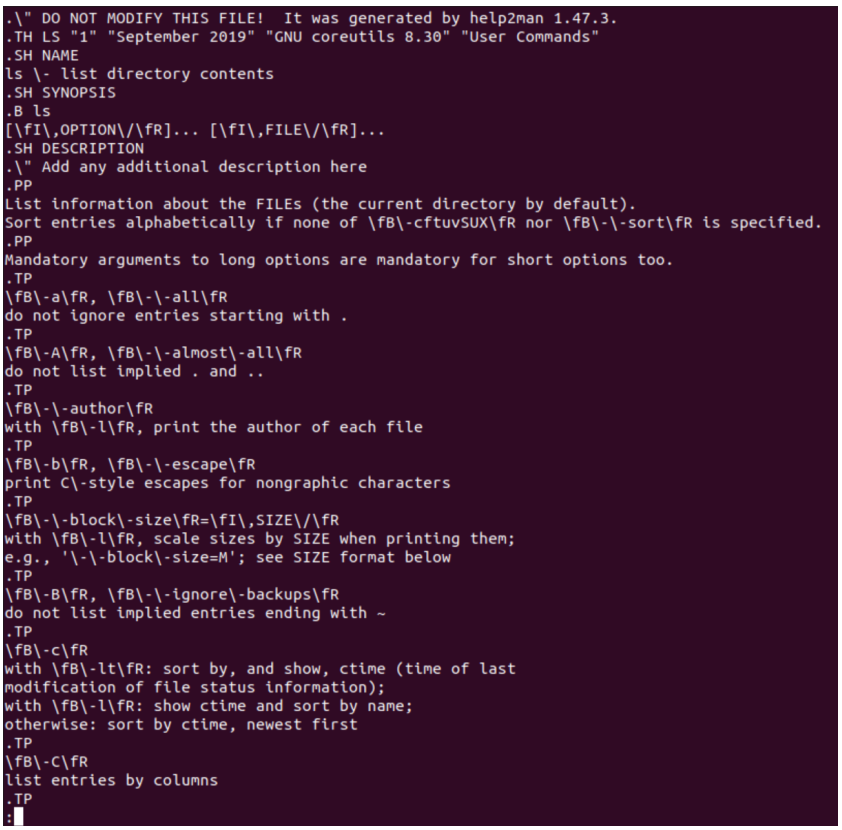

Рис. 2.7: Пример использования второй программы

Третья программа:

Используя встроенную переменную $RANDOM, напишите командный файл, генерирующий случайную последовательность букв латинского алфавита. Учтите, что $RANDOM выдаёт псевдослучайные числа в диапазоне от 0 до 32767. Код третьей программы. (рис. 2.8).

```
$ random.sh
#!/bin/bash

generate_rand_lett() {
    alphabet="abcdefghijklmnopqrstuvwxyz"
    random_index=$((RANDOM % ${#alphabet}))
    random_letter="${alphabet:$random_index:1}"
    echo "$random_letter"
}

random_seq=""
for ((i=0; i<10; i++)); do
    random_letter=$(generate_rand_lett)
    random_seq="$random_seq$random_letter"
done

echo "Случайная последовательность: $random_seq"
```

Рис. 2.8: Код третьей программы

Пример использования третьей программы. (рис. 2.9).

```
guldyaev-tikhon@guldyaevtikhon:~/13$ ./random.sh
Случайная последовательность: jnsunjwwhs
guldyaev-tikhon@guldyaevtikhon:~/13$ ./random.sh
Случайная последовательность: xsyvnlfqfa
guldyaev-tikhon@guldyaevtikhon:~/13$ ./random.sh
Случайная последовательность: hwjqxvhgxc
guldyaev-tikhon@guldyaevtikhon:~/13$ ./random.sh
Случайная последовательность: klrugvnfhb
guldyaev-tikhon@guldyaevtikhon:~/13$ ./random.sh
Случайная последовательность: ildyulnvbu
guldyaev-tikhon@guldyaevtikhon:~/13$
```

Рис. 2.9: Пример использования третьей программы

3. Выводы

Я изучил основы программирования в оболочке ОС UNIX. Научился писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

4. Ответы на контрольные вопросы

1. Найдите синтаксическую ошибку в следующей строке: `while [$1 != "exit"]`

   Синтаксическая ошибка в данной строке заключается в отсутствии пробелов вокруг символа [и отсутствие кавычек вокруг переменной $1. Верное написание строки будет следующим:

   ```
   while [ "$1" != "exit" ]
   ```

2. Как объединить (конкатенация) несколько строк в одну?

   - Использование оператора конкатенации +=
   - Использование оператора конкатенации внутри кавычек

     ```
     concatenated_string="${string1}${string2}"
     ```

   - Использование команды printf с форматированием

     ```
     concatenated_string=$(printf "%s%s" "$string1" "$string2")
     ```

3. Найдите информацию об утилите seq. Какими иными способами можно реализовать её функционал при программировании на bash?

   Утилита seq в Bash используется для генерации числовых последовательностей. Ее функциональность состоит в создании последовательности чисел от начального значения до конечного значения с заданным шагом.

   Некоторые альтернативные способы реализации функционала seq в программировании на Bash включают:

   - Использование цикла for с инкрементом

- Использование цикла while с инкрементом
- Использование массива и цикла for

4. Какой результат даст вычисление выражения `$((10/3))`?

   Выражение `$((10/3))` в Bash будет вычислено как деление 10 на 3, а результат будет округлен вниз до целого числа. В данном случае результат будет равен 3.

5. Укажите кратко основные отличия командной оболочки zsh от bash.

   Ниже приведены основные отличия между командными оболочками zsh и bash:

   1. Синтаксис и расширенные возможности: Zsh предлагает более расширенный и мощный синтаксис команд, включая автозаполнение (Tab completion) с подсказками, расширенные шаблоны и множество встроенных функций.
   2. Автозаполнение и автодополнение: Zsh имеет более продвинутую систему автозаполнения, которая может предложить варианты завершения команд и аргументов, основываясь на истории команд, путях файловой системы и других контекстных данных.
   3. Поддержка смены рабочей директории: В Zsh можно быстро переключаться между рабочими директориями без необходимости указывать полный путь.
   4. Настройка и темы оформления: Zsh предлагает более гибкую и мощную систему настройки и настройку тем оформления для командной строки.
   5. Лучшая обработка ошибок: Zsh обрабатывает ошибки более информативно, позволяя легче понять, что пошло не так при выполнении команд.
   6. Удобные и продвинутые возможности истории команд: Zsh предоставляет расширенные функции работы с историей команд, включая поиск, фильтрацию, удаление дубликатов и другие возможности.
   7. Скорость выполнения команд: В некоторых сценариях Zsh может быть быстрее в выполнении команд и операций, чем Bash.

   Однако стоит отметить, что Bash является более широко распространенной командной оболочкой и поставляется по умолчанию во многих системах Linux. Отличия между Zsh

и Bash могут быть важны для опытных пользователей, которые хотят настроить свою командную оболочку под свои нужды и предпочтения.

6. Проверьте, верен ли синтаксис данной конструкции `for ((a=1; a <= LIMIT; a++))`

   Синтаксис данной конструкции в целом верен, но вам необходимо определить значение переменной LIMIT перед использованием цикла.

7. Сравните язык bash с какими-либо языками программирования. Какие преимущества у bash по сравнению с ними? Какие недостатки?

   Преимущества языка bash:

   - Простота использования и быстрота разработки для автоматизации задач командной строки и скриптования.
   - Встроенная поддержка многих системных утилит и команд операционной системы.
   - Широкая доступность и предустановленность в большинстве Unix-подобных систем.
   - Удобное управление файлами, потоками и процессами через конвейеры и перенаправления.
   - Интеграция с системными сервисами и инструментами, такими как cron, systemd и другими.

   Недостатки языка bash:

   - Ограниченные возможности для разработки сложных и масштабируемых приложений.
   - Нет поддержки типов данных, структур данных и объектно-ориентированного программирования.
   - Неэффективность при выполнении сложных вычислений и больших объемов данных.
   - Отсутствие расширенной обработки ошибок и исключений.

- Ограниченные возможности для создания графического интерфейса или веб-приложений.

Сравнивая bash с языками программирования C и Python, следует отметить, что bash является специализированным языком для работы с командной строкой и автоматизации системных задач, в то время как C и Python более общепринятые языки программирования с широким спектром применения. C обеспечивает более низкоуровневый доступ к системным ресурсам и эффективность, но требует более глубокого понимания и имеет более сложный синтаксис. Python обладает более высоким уровнем абстракции, богатым набором библиотек и инструментов, поддерживает объектно-ориентированное программирование и широко применяется для разработки веб-приложений, научных вычислений и автоматизации задач.

В итоге, bash предоставляет простой и удобный способ для автоматизации системных задач и работы с командной строкой, но ограничен в возможностях программирования по сравнению с C и Python, которые предлагают более общие инструменты и гибкость для разработки различных видов приложений.

Список литературы

https://www.google.ru

https://chat.openai.com/chat