Лабораторная работа №12

Программирование в командном процессоре ОС UNIX. Расширенное программирование

Мажитов Магомед Асхабович

Содержание

1	Цель работы	5
2	Задачи	6
3	Ход работы	8
4	Вывод	11
5	Контрольные вопросы.	12

Список иллюстраций

3.1	Код 1 скрипта															8
3.2	Работа скрипта															9
3.3	Код 2 скрипта															9
3.4	Работа скрипта															10
3.5	Код 3 скрипта															10
3.6	Работа скрипта															10

Список таблиц

1 Цель работы

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

2 Задачи

- 1. 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.
- 2. 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.
- 3. Используя встроенную переменную \$RANDOM, напишите командный файл, генерирующий случайную последовательность букв латинского алфавита. Учтите, что \$RANDOM выдаёт псевдослучайные числа в диапазоне от 0 до

3 Ход работы

1. Написал 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). (рис. 3.1)

```
#!/bin/bash
LOCKFILE="./lock.file"
exec {fn}>$LOCKFILE

if test -f "$LOCKFILE"
then

while
        [ 1 = 1 ]
        do
        if flock -n ${fn}
        then
            echo "Файл заблокирован"
            sleep 3
            echo "Файл разблокирован"
            flock -u ${fn}
        else
            echo "Файл заблокирован"
            sleep 3
            flock -u ${fn}
        else
            echo "Файл заблокирован"
            sleep 3
            flock -u $fn}
```

Рис. 3.1: Код 1 скрипта

Запустил скрипт. (рис. 3.2)

```
[mamazhitov@fedora lab12]$ chmod +x script1
[mamazhitov@fedora lab12]$ ./script1
Файл заблокирован
Файл разблокирован
Файл заблокирован
^C
[mamazhitov@fedora lab12]$
```

Рис. 3.2: Работа скрипта

2. Реализовал команду man с помощью командного файла. Командный файл получает в виде аргумента командной строки название команды и в виде результата выдает справку об этой команде или сообщение об отсутствии справки, если соответствующего файла нет в каталоге man1.(рис. 3.3)

```
#1/bin/bash
command=""

while getopts :n: opt
do
case $opt in
n)command="$OPTARG";;
esac
done

if test -f "/usr/share/man/man1/$command.1.gz"
then less /usr/share/man/man1/$command.1.gz
else
echo "no such command"
fi
```

Рис. 3.3: Код 2 скрипта

Запустил скрипт.(рис. 3.4)

```
BASH_BUILTINS(1)
                                                                                                                                                                                                                                                              BASH_BUILTINS(1)
                                                                                                                   General Commands Manual
ESC[1mNAMEESC[0m
                            bash, :, ., [, alias, bg, bind, break, builtin, caller, cd, command, compgen, complete, compopt, continue, declare, dirs, disown, echo, en-
                            able, eval, exec, exit, export, false, fc, fg, getopts, hash, help, history, jobs, kill, let, local, logout, mapfile, popd, printf, pushd,
                            pwd, read, readonly, return, set, shift, shopt, source, suspend, test,
                            times, trap, true, type, typeset, ulimit, umask, unalias, unset, wait bash built-in commands, see <a href="ESC">ESC</a>[Imbash</a>[22m(1)
ESC[1mBASH BUILTIN COMMANDSESC[0m
                            Unless otherwise noted, each builtin command documented in this section as accepting options preceded by <a href="ESC">ESC</a>[22m</a>
        signify the end of the
osignity the end of the social stress of the stress of the
                        and process arguments beginning
                            with ESC[1m- ESC[22mwithout requiring ESC[1m--ESC[22m. Other builtins the
at accept arguments but
 /usr/share/man/man1/cd.1.03
```

Рис. 3.4: Работа скрипта

3. Используя встроенную переменную \$RANDOM, напишите командный файл, генерирующий случайную последовательность букв латинского алфавита. (рис. 3.5)



Рис. 3.5: Код 3 скрипта

Запустил скрипт.(рис. 3.6)



Рис. 3.6: Работа скрипта

4 Вывод

Мы научились писать более сложные командные файлы.

5 Контрольные вопросы.

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

\$1.

Так же между скобками должны быть пробелы. В противном случае скобки и рядом стоящие символы будут восприниматься как одно целое

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

```
cat file.txt | xargs | sed -e 's/\. /.\n/g'
```

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

seq - выдает последовательность чисел. Реализовать ее функционал можно командой for n in {1..5} do done

4: Какой результат даст вычисление выражения \$((10/3))?

3

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

Zsh очень сильно упрощает работу. Но существуют различия. Например, в zsh после for обязательно вставлять пробел, нумерация массивов в zsh начинается с 1 (что не особо удобно на самом деле). Если вы собираетесь писать скрипт, который легко будет запускать множество разработчиков, то я рекомендуется Bash. Если скрипты вам не нужны - Zsh (более простая работа с файлами, например)

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

Верен

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

Bash позволяет очень легко работать с файловой системой без лишних конструкций (в отличи от обычного языка программирования). Но относительно обычных языков программирования bash очень сжат. Тот же Си имеет гораздо более широкие возможности для разработчика.