

Лабораторная работа №12

**Программирование в командном процессоре ОС UNIX. Расширенное
программирование**

Извекова Мария Петровна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	8
4	Выводы	11
	Список литературы	12

Список иллюстраций

3.1	рис. 1	8
3.2	рис. 2	9
3.3	рис. 3	9
3.4	рис. 4	9
3.5	рис. 5	10
3.6	рис. 6	10
3.7	рис. 7	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

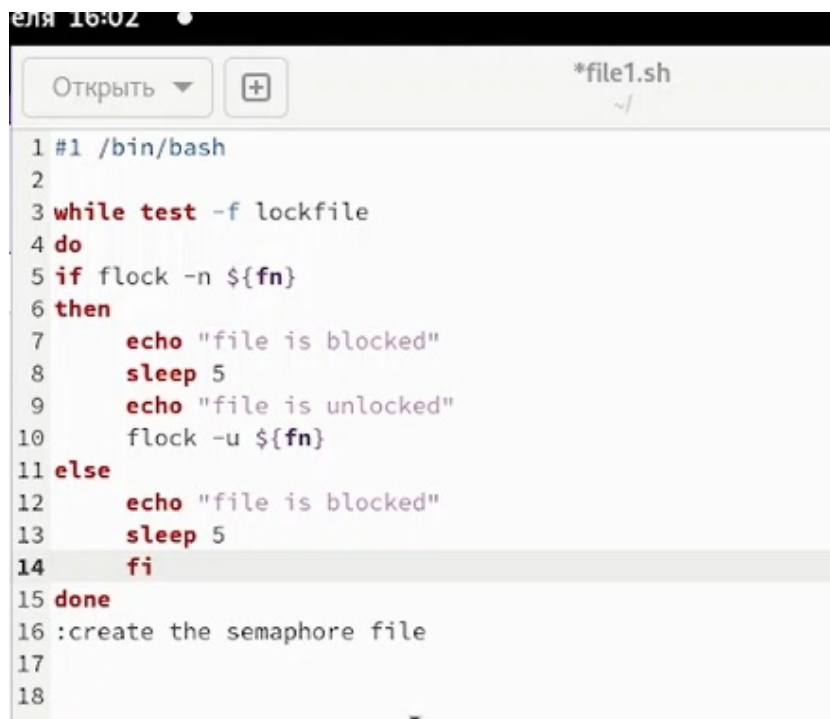
2 Задание

1. Написать командный файл, реализующий упрощённый механизм семафоров. Командный файл должен в течение некоторого времени t_1 дожидаться освобождения ресурса, выдавая об этом сообщение, а дождавшись его освобождения, использовать его в течение некоторого времени $t_2 < t_1$, 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

32767

3 Выполнение лабораторной работы

1. Пишем команду, которая определяет, файл является в открытом доступе (unlocked) или в заблокированном (blocked)



```
1 #! /bin/bash
2
3 while test -f lockfile
4 do
5     if flock -n ${fn}
6     then
7         echo "file is blocked"
8         sleep 5
9         echo "file is unlocked"
10        flock -u ${fn}
11    else
12        echo "file is blocked"
13        sleep 5
14    fi
15 done
16 :create the semaphore file
17
18
```

Рис. 3.1: рис. 1

2. Вывод команды. Он у меня не получился, но код должен выводить выражения, чтобы понять, какой доступ у файла


```

[marieizvekova@fedora ~]$ gedit file1.sh
[marieizvekova@fedora ~]$ $
bash: $: команда не найдена...
[marieizvekova@fedora ~]$ bash file1.sh
file1.sh: строка 3: lockfile./lock.file: Нет такого файла или каталога
file1.sh: строка 4: fn: неоднозначное перенаправление
[marieizvekova@fedora ~]$ gedit file1.sh
[marieizvekova@fedora ~]$ bash file1.sh
file1.sh: строка 3: lockfiles./lock.file: Нет такого файла или каталога
file1.sh: строка 4: fn: неоднозначное перенаправление
[marieizvekova@fedora ~]$ gedit file1.sh
[marieizvekova@fedora ~]$ bash file1.sh
file1.sh: строка 3: lockfiles./lock.file: Нет такого файла или каталога
file1.sh: строка 4: fn: неоднозначное перенаправление
[marieizvekova@fedora ~]$

```

Рис. 3.2: рис. 2

3. Следующий код должен выводить опционал каждой команды



```

1 #!/bin/bash
2
3 a=$1
4 if test -f "/usr/share/man/man1/$a.1.gz"
5 then less /usr/share/man/man1/$a.1.gz
6 else
7 echo "not found"
8 fi
9

```

Рис. 3.3: рис. 3

4. Команда, которая это обрабатывает



```

[marieizvekova@fedora ~]$ gedit file2.sh
[marieizvekova@fedora ~]$ bash file2.sh mkdir

```

Рис. 3.4: рис. 4

5. Что получаем

```

marieizvekova@fedora:~ — bash file2.sh mkdir
MKDIR(1)                                User Commands                                MKDIR(1)

ESC[1mNAMEESC[0m
mkdir - make directories

ESC[1mSYNOPSISESC[0m
ESC[1mmkdir ESC[22m[ESC[4mOPTIONESC[24m]... ESC[4mDIRECTORYESC[24m]...

ESC[1mDESCRIPTIONESC[0m
Create the DIRECTORY(ies), if they do not already exist.

Mandatory arguments to long options are mandatory for short options too.

ESC[1m-mESC[22m, ESC[1m--modeESC[22m=ESC[4mMODEESC[0m
set file mode (as in chmod), not a=rwx - umask

ESC[1m-pESC[22m, ESC[1m--parentsESC[0m
no error if existing, make parent directories as needed, with
their file modes unaffected by any ESC[1m-m ESC[22moption.

ESC[1m-vESC[22m, ESC[1m--verboseESC[0m
print a message for each created directory

/usr/share/man/man1/mkdir.1.gz

```

Рис. 3.5: рис. 5

6. Последний код в случайном порядке должен выводить буквы

```

file3.sh
1 #!/bin/bash
2
3 a=$1
4 for ((i=0; i<a; i++))
5 do
6   ((char=$RANDOM26+1))
7   case $char in
8     1) echo n a;; 2) echo n b;; 3) echo n c;; 4) echo n d;; 5) echo n e;;
9     6) echo n f;; 7) echo n g;; 8) echo n h;; 9) echo n i;; 10) echo n j;;
10    11) echo n k;; 12) echo n l;; 13) echo n m;; 14) echo n n;; 15) echo n o;;
11    16) echo n p;; 17) echo n q;; 18) echo n r;; 19) echo n s;; 20) echo n t;;
12    21) echo n u;; 22) echo n v;; 23) echo n w;; 24) echo n x;; 25) echo n y;;
13    26) echo n z;;
14 done

```

Рис. 3.6: рис. 6

7. Итог этой команды

```

marieizvekova@fedora ~]$ bash file3.sh 9
aaaaaaaaa
marieizvekova@fedora ~$

```

Рис. 3.7: рис. 7

4 Выводы

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

Список литературы