

Отчёт по лабораторной работе № 12

Королёв Иван Андреевич

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	8
4	Выполнение лабораторной работы	9
4.1	Написать командный файл, реализующий упрощённый механизм семафоров	9
4.2	Реализовать команду tap с помощью командного файла	10
4.3	Используя встроенную переменную \$RANDOM, напишите командный файл	12
5	Выводы	14
6	Ответ на контрольные вопросы	15

Список иллюстраций

4.1	file1.sh	9
4.2	file1.sh	10
4.3	file2.sh	11
4.4	file2.sh	11
4.5	file2.sh	12
4.6	file2.sh	12
4.7	file3.sh	13
4.8	file3.sh	13

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 Теоретическое введение

Командный процессор (командная оболочка, интерпретатор команд shell) — это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера. В операционных системах типа UNIX/Linux наиболее часто используются следующие реализации командных оболочек: – оболочка Борна (Bourne shell или sh) — стандартная командная оболочка UNIX/Linux, содержащая базовый, но при этом полный набор функций; – C-оболочка (или csh) — надстройка на оболочкой Борна, использующая C-подобный синтаксис команд с возможностью сохранения истории выполнения команд; – оболочка Корна (или ksh)—напоминает оболочку C,но операторы управления программой совместимы с операторами оболочки Борна; – BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек C и Корна (разработка компании Free Software Foundation). POSIX (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ. Стандарты POSIX разработаны комитетом IEEE (Institute of Electrical and Electronics Engineers) для обеспечения совместимости различных UNIX/Linux-подобных операционных систем и переносимости прикладных программ на уровне исходного кода. POSIX-совместимые оболочки разработаны на базе оболочки Корна.

4 Выполнение лабораторной работы

4.1 Написать командный файл, реализующий упрощённый механизм семафоров

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). (рис. 4.1), (рис. 4.2)

```
[iakorolev@fedora lab12]$ bash file1.sh
File is blocked
File is unlocked
File is blocked
File is unlocked
File is blocked
File is unlocked
File is blocked
^C
```

Рис. 4.1: file1.sh

```

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

```

Рис. 4.2: file1.sh

4.2 Реализовать команду man с помощью командного файла

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. (рис. 4.3), (рис. 4.4), (рис. 4.5), (рис. 4.6)

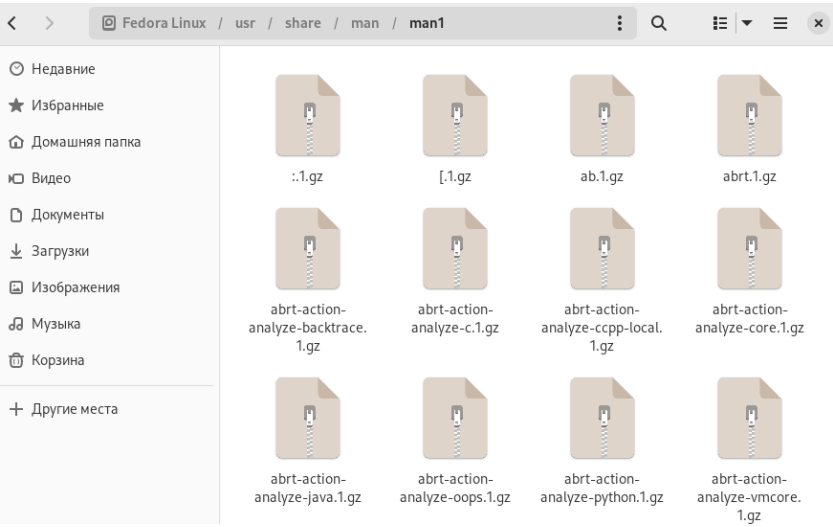


Рис. 4.3: file2.sh

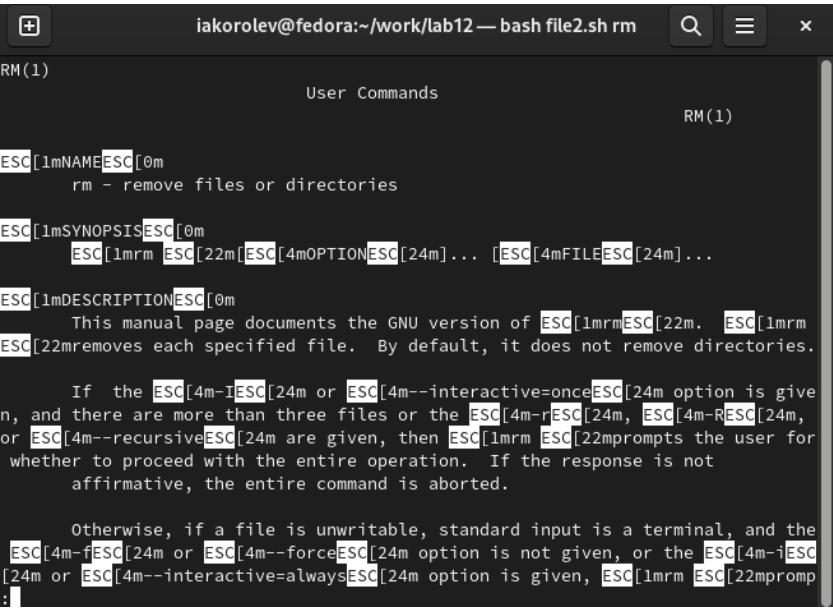
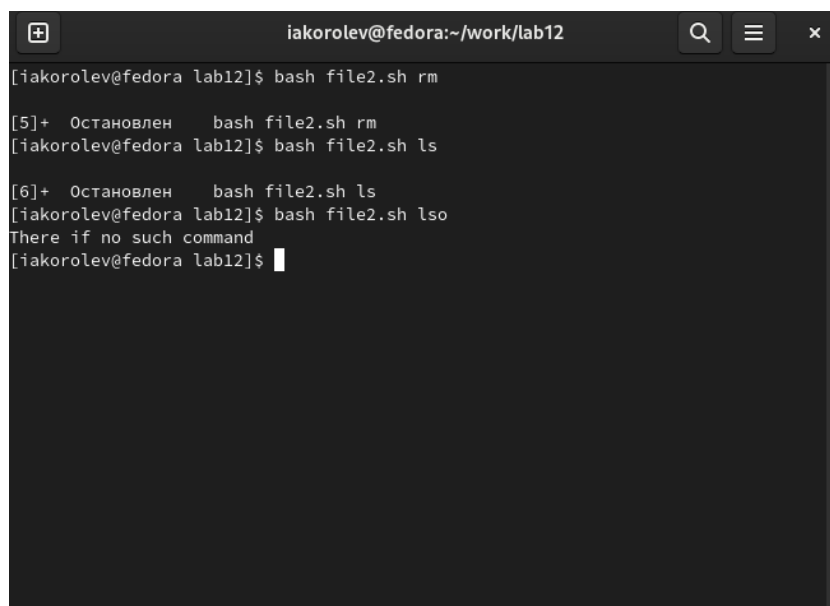


Рис. 4.4: file2.sh



```
iaakorolev@fedora:~/work/lab12
[iaakorolev@fedora lab12]$ bash file2.sh rm
[5]+ Остановлен bash file2.sh rm
[iaakorolev@fedora lab12]$ bash file2.sh ls
[6]+ Остановлен bash file2.sh ls
[iaakorolev@fedora lab12]$ bash file2.sh lso
There if no such command
[iaakorolev@fedora lab12]$
```

Рис. 4.5: file2.sh

```
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 "There if no such command"
8 fi
```

Рис. 4.6: file2.sh

4.3 Используя встроенную переменную \$RANDOM, напишите командный файл

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. (рис. 4.7), (рис. 4.8)

```
[iakorolev@fedora lab12]$ bash file3.sh 25
axtroesekpqpexymzhpvhfrih
[iakorolev@fedora lab12]$ bash file3.sh 2145
rlzfofuncvymrxoreoqgmkaioxxiqjjhlfhhdgeogqaobyqmnmkqakbotezusxtxauxeiovlrukzib
geaojogomkhpuevkjrwslfhrhziuyilpxhoyiazoneqehowioqdbwqhazuwcgeldkdjismwmfiesdmv
vradrrheijbwndqztlvkwugeyteparwramsundklbtctxhgrfgmazrcbptfbyoxramidaaevimggad
pjsumnzkiwhlbecxfgvrrslvvzykuhjeymmvglactwsastrmjfjhlhuzourexhzzcwietfskivuqvd
hyhxsppdpusohmdsurxxzrupacjctlsghgpmdzhlviyodaaffxjjtjjashpronrbznfhdcxjvcvtkin
jhchqmmzyvlbgrinuzcyoixbmaggjbrrrhdoiywtvqhpponpeuocyrrpfdqtqfomkxntqunsmkzgbib
vfjkgjsjucwwnoocmsqgzwwrncrohvhuacltnonutraqmlpfegilrpwnhhpiuldziesyshoelhyemsp
exjedyhqsqurpkcdilwrjtslnibxjwnlikuxpkcimpsemknzdyaqfvonfgixurnkjnbndgrfckzfxjp
yvvubakwtnojzsvcllegepkebutshgxlncuycgknaefoatwscskdkxfqxqethiwmuehosfoeitgbpfd
nfelsnslyyeuhsqzpmxzvrodppzlimekmrunyghdrprsyadoedwqxcvafpskoidrfhmpyvqtkkywvpm
mgasullftqlqrxzaxbtlywselxhlumfuenajguqfuwvobmnnszvelmzyggadlobznquyxhjrkpzcztz
hzgykegbfodtjvsfezmlncpwyqqiddvyqshxxuhddwleddrsvoseqegqtcpcfwalgtlidiuutpgbkgedc
bpociljsoewklncxrrfyrvdccyyusoyaqspfyplfswiuxeawnbmcxpwuhukjqbdzwglalbclyxkxrkvg
tmkwpbhyfixoscwyxfhxrllwshfqthimkadbhwevixrnrpcxqyginkoxiszavavmytoblynxdqzfcfgh
nsmxucjjqzfdyicklfcwlayncyxtnwewcguhofaktmwwjwksnhhztmnrstajyxupiajgxyuhhhovtvd
nibwailltzzsosflkliwgewmnzhlqrzcrpstvtutmjebzsnjyzzjwgkzladjmzvlfwrrwrnpjpsjuklvh
bwaxdlebonuzsebtwkelbspzewuaxlsiqpumwkjxutzbabpdzjijivyjjkrzfxcarqnpqrcanukss
avykfsqhrhyuirvtputyhykmutcnqxbtmcwjexcmhradbxbmvqcezolktxakgvosrtsasauzwybzqfppi
```

Рис. 4.7: file3.sh

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

Рис. 4.8: file3.sh

5 Выводы

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

6 Ответ на контрольные вопросы

1. В данной строчке допущены следующие ошибки: не хватает пробелов после первой скобки [и перед второй скобкой] выражение \$1 необходимо взять в “”, потому что эта переменная может содержать пробелы. Таким образом, правильный вариант должен выглядеть так: `while [“$1” != “exit”]`
2. Чтобы объединить несколько строк в одну, можно воспользоваться несколькими способами: Первый: `VAR1=“Hello,” VAR2=“ World” VAR3=“VAR1VAR2”`
`echo “VAR3” : Hello, World : VAR1 = “Hello,”VAR1+ =`
`“World”echo“VAR1”` Результат: Hello, World
3. Команда `seq` в Linux используется для генерации чисел от ПЕРВОГО до ПОСЛЕДНЕГО шага INCREMENT. Параметры: `seq LAST`: если задан только один аргумент, он создает числа от 1 до LAST с шагом шага, равным 1. Если LAST меньше 1, значение `is` не выдает. `seq FIRST LAST`: когда заданы два аргумента, он генерирует числа от FIRST до LAST с шагом 1, равным 1. Если LAST меньше FIRST, он не выдает никаких выходных данных. `seq FIRST INCREMENT LAST`: когда заданы три аргумента, он генерирует числа от FIRST до LAST на шаге INCREMENT. Если LAST меньше, чем FIRST, он не производит вывод. `seq -f «FORMAT» FIRST INCREMENT LAST`: эта команда используется для генерации последовательности в форматированном виде. FIRST и INCREMENT являются необязательными. `seq -s «STRING» ПЕРВЫЙ ВКЛЮЧЕНО`: Эта команда используется для STRING для разделения чисел. По умолчанию это значение равно /n. FIRST и INCREMENT являются необязательными. `seq -w FIRST INCREMENT LAST`: эта команда используется для

выравнивания ширины путем заполнения начальными нулями. FIRST и INCREMENT являются необязательными.

4. Результатом данного выражения $\$(10/3)$ будет 3, потому что это целочисленное деление без остатка.
5. Отличия командной оболочки zsh от bash: В zsh более быстрое автодополнение для cd с помощью Tab В zsh существует калькулятор zcalc, способный выполнять вычисления внутри терминала В zsh поддерживаются числа с плавающей запятой В zsh поддерживаются структуры данных «хэш» В zsh поддерживается раскрытие полного пути на основе неполных данных В zsh поддерживается замена части пути В zsh есть возможность отображать разделенный экран, такой же как разделенный экран vim
6. `1 for ((a=1; a <= LIMIT; a++))`

`for ((a=1; a <= LIMIT; a++))` синтаксис данной конструкции верен, потому что, используя двойные круглые скобки, можно не писать \$ перед переменными ().

7. Преимущества скриптового языка bash:

- Один из самых распространенных и ставится по умолчанию в большинстве дистрибутивах Linux, MacOS
- Удобное перенаправление ввода/вывода
- Большое количество команд для работы с файловыми системами Linux
- Можно писать собственные скрипты, упрощающие работу в Linux Недостатки скриптового языка bash:
- Дополнительные библиотеки других языков позволяют выполнить больше действий
- Bash не является языком общего назначения
- Утилиты, при выполнении скрипта, запускают свои процессы, которые, в свою очередь, отражаются на скорости выполнения этого скрипта

- Скрипты, написанные на `bash`, нельзя запустить на других операционных системах без дополнительных действий