

Отчёт по лабораторной работе №14

Программирование в командном процессоре ОС UNIX.

Сейдалиев Тагьетдин Ровшенович

Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
3	Вывод	7
4	Контрольные вопросы	8

List of Figures

2.1	Задание 1	5
2.2	Задание 2	6
2.3	Задание 3	6

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

1. Написали командный файл, реализующий упрощённый механизм семафоров. Командный файл в течение некоторого времени t_1 дожидается освобождения ресурса, выдавая об этом сообщение, а дождавшись его освобождения, использует его в течение некоторого времени $t_2 < t_1$, также выдавая информацию о том, что ресурс используется соответствующим командным файлом (процессом). Запустили 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. Доработали программу так, чтобы имелась возможность взаимодействия трёх и более процессов.

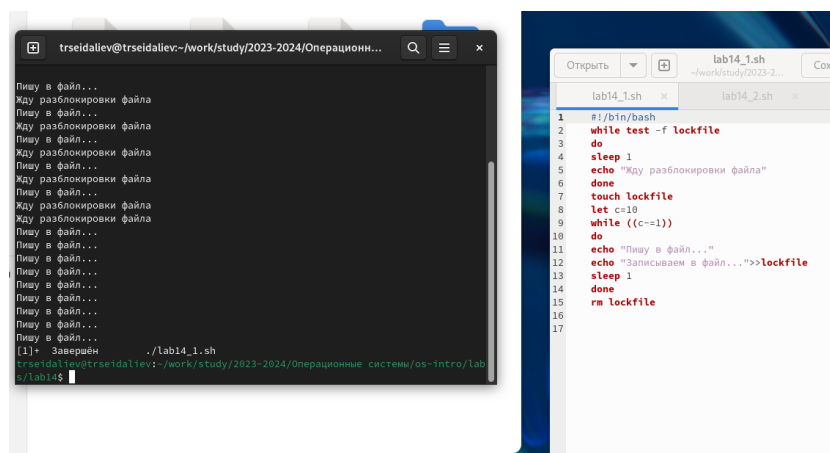
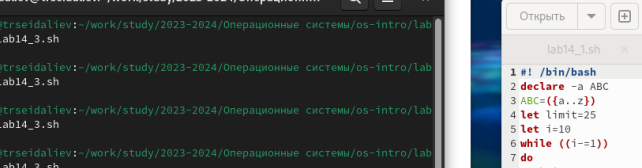


Figure 2.1: Задание 1

2. Реализовали команду `map` с помощью командного файла. Изучили содержи-

```
trseidaliev@trseidaliev:~/work/study/2023-2024/Операционн...  
ESC[4mESC[24m(1) User Commands  
ESC[4mESC[24m(1)  
ESC[1mNAMEESC[0m  
cp - copy files and directories  
ESC[1mSYNOPSISESC[0m  
ESC[1mcp [mcp [22mESC[4mOPTIONESC[24m)... [ESC[4m-TEESC[24m] ESC[4mSOURCE  
ESC[24mESC[4mDESTESC[0m  
ESC[1mcp [mcp [22mESC[4mOPTIONESC[24m)... ESC[4mSOURCEESC[24m... ESC[4mDIR  
ECTORVESC[0m  
ESC[1mcp [mcp [22mESC[4mOPTIONESC[24m)... ESC[4m-TEESC[24m] ESC[4mDIRECTORY  
ESC[24mESC[4mSOURCEESC[24m...  
ESC[1mDESCRIPTIONESC[0m  
Copy SOURCE to DEST, or multiple SOURCE(s) to DIRECTORY.  
  
Mandatory arguments to long options are mandatory for short options  
too.  
  
ESC[1m-ESC[22mESC[0mESC[1m--archiveESC[0m  
same as ESC[1m-dr --preserveESC[22mESC[4mESC[0m  
  
/usr/share/man/man1/cp.1.gz
```

3. Используя встроенную переменную \$RANDOM, написали командный файл, генерирующий случайную последовательность букв латинского алфавита



```
trseidaliev@trseidaliev:~/work/study/2023-2024/Операционни...
trseidaliev@trseidaliev:~/work/study/2023-2024/Операционные системы/os-intro/lab
s/lab14$ ./lab14_3.sh
okwrvvoqch
trseidaliev@trseidaliev:~/work/study/2023-2024/Операционные системы/os-intro/lab
s/lab14$ ./lab14_3.sh
utavwoofe
trseidaliev@trseidaliev:~/work/study/2023-2024/Операционные системы/os-intro/lab
s/lab14$ ./lab14_3.sh
srwokuynv
trseidaliev@trseidaliev:~/work/study/2023-2024/Операционные системы/os-intro/lab
s/lab14$ ./lab14_3.sh
kurwkgdsdr
trseidaliev@trseidaliev:~/work/study/2023-2024/Операционные системы/os-intro/lab
s/lab14$ ./lab14_3.sh
wfpafomf
trseidaliev@trseidaliev:~/work/study/2023-2024/Операционные системы/os-intro/lab
s/lab14$
```

6

3 Вывод

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

4 Контрольные вопросы

1. Найдите синтаксическую ошибку в следующей строке: `while [$1 != "exit"]` Ответ: Правильный вариант: `while ["$1" != " exit"]`
2. Как объединить (конкатенация) несколько строк в одну? Ответ: Объединение нескольких строк в одну в Bash происходит с помощью символа `'\ '`
3. Найдите информацию об утилите `seq`. Какими иными способами можно реализовать её функционал при программировании на `bash`? Ответ для примера: В Linux имеется программа `seq`, которая воспринимает в качестве аргументов два числа и выдает последовательность всех чисел, расположенных между заданными. С помощью этой команды можно заставить `for` в `bash` работать точно так же, как аналогичный оператор работает в обычных языках программирования. Для этого достаточно записать цикл `for` следующим образом:

```
for a in $( seq 1 10 ) ; do
catfile_$a
done
```

Эта команда выводит на экран содержимое 10-ти файлов:
“file_1”, ..., “file_10”.

4. Какой результат даст вычисление выражения `$((10/3))`? Ответ: Так как это целочисленное деление, то произойдет округление в сторону ближайшего числа, и выведется 3. $10/3 = 3$.

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

По размеру Bash больше Zsh. Zsh и Bash предлагают сходный функционал. Обе имеют программируемое дополнение (хотя у Zsh оно появилось раньше), встроенные команды и функции для создания скриптов. У Zsh также в запасе есть несколько собственных хитростей, например, расширенная подстановка имени файла, которая превращает команду поиска find почти что в ненужное излишество. Включение в путь ** означает соответствие любому символу, включая разделитель - слэш, поэтому **/*.jpg касается всех файлов *.jpg в текущей директории и в любых поддиректориях. Мало того, сюда также включаются права доступа к файлу, владелец, тип или отметка времени – большинство опций, предусмотренных find. Например, можно использовать ls -l /**/bin/*(s) для вывода списка всех setuid-файлов в /bin, /usr/bin и /usr/local/bin. При наборе имени директории в командной строке Zsh переключается на эту директорию. Выполнение скриптов в Zsh основном быстрее, чем в Bash – по большей части примерно на 20% – однако Zsh разработан для интерактивного пользования. В Zsh расширенная подстановка имени файла и более развитая опция дополнения..

6. Проверьте, верен ли синтаксис данной конструкции for ((a=1; a <= LIMIT; a++)) Ответ: В bash для оператора цикла for существует другая конструкция.

Пример:

```
for A in Ai Bi Ci do
echo A
done
```

на терминал будет выведено :

```
Ai Bi Ci
```

7. Сравните язык bash с какими-либо языками программирования. Какие

преимущества у `bash` по сравнению с ними? Какие недостатки? Ответ: Вначале был Bourne Shell (`sh`), его написал Стивен Борн для Bell Labs Research Unix. Bash – это Bourne Again Shell (Снова Оболочка Борна), который, к счастью, редко используется. Почти все современные дистрибутивы Linux используют Bash в качестве оболочки по умолчанию, и это превращает Bash в фактический стандарт, с которым сравниваются все остальные. Дело не в малом размере Bash, и не в скорости. По размеру Bash больше некоторых оболочек, кроме одной: Sash, которая не использует библиотек и имеет несколько дополнительных встроенных команд. Bash также и не самая быстрая оболочка, однако большинству пользователей это неважно, ибо подлинно важна его гибкость. Bash обладает некоторыми функциями, превосходящими стандарт POSIX, хотя при желании можно добиться от него и POSIX-поведения. Если запустить Bash командой `sh`, с опцией командной строки `-posix` или при установленной переменной окружения `POSIXLY_CORRECT`, Bash будет работать как стандартная оболочка POSIX. При запуске через `sh`, Bash по возможности пытается работать как исходная оболочка Борна, но лишь в тех ситуациях, когда это не вступит в конфликт со стандартом POSIX.