

Отчёт по лабораторной работе №14

Программирование в командном процессоре ОС UNIX.

Гейдарова Назиля Рамиль кызы

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Вывод	9
4	Контрольные вопросы	10

Список иллюстраций

2.1	Задание 1	6
2.2	Задание 2	7
2.3	Задание 3	8

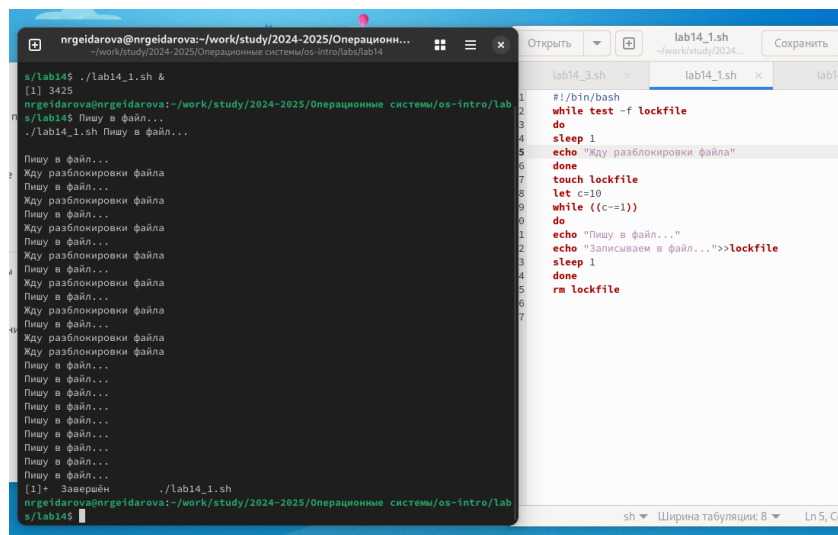
Список таблиц

1 Цель работы

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

2 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.

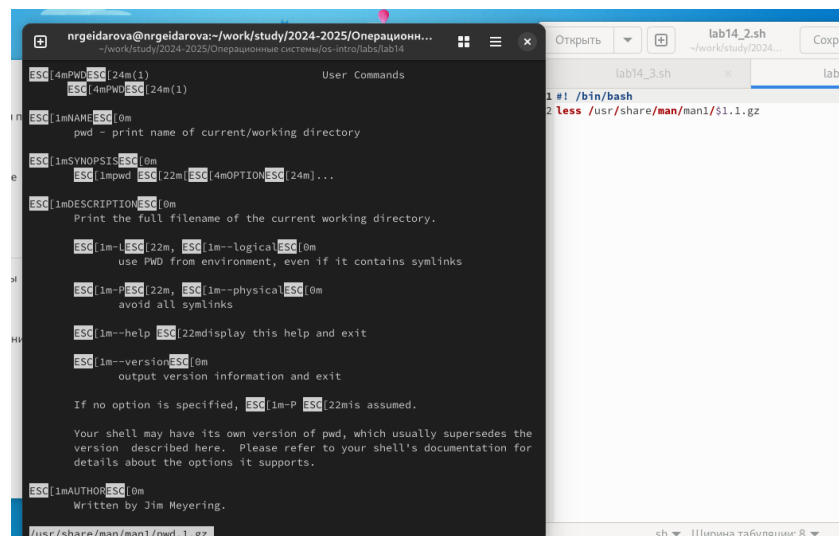


```
s/lab14$ ./lab14_1.sh &
[1] 3425
nrgeldarova@nrgeldarova:~/work/study/2024-2025/Операционные системы/os-intro/lab14
s/lab14$ ПИШУ в файл...
./lab14_1.sh ПИШУ в файл...
ПИШУ в файл...
ЖДУ разблокировки файла
ПИШУ в файл...
ЖДУ разблокировки файла
ПИШУ в файл...
ЖДУ разблокировки файла
ПИШУ в файл...
ЖДУ разблокировки файла
ПИШУ в файл...
ЖДУ разблокировки файла
ПИШУ в файл...
ЖДУ разблокировки файла
ПИШУ в файл...
ЖДУ разблокировки файла
ПИШУ в файл...
ПИШУ в файл...
ПИШУ в файл...
ПИШУ в файл...
ПИШУ в файл...
ПИШУ в файл...
ПИШУ в файл...
[1]+  Завершён ./lab14_1.sh
nrgeldarova@nrgeldarova:~/work/study/2024-2025/Операционные системы/os-intro/lab14
s/lab14$
```

```
#!/bin/bash
while test -f lockfile
do
sleep 1
echo "ЖДУ разблокировки файла"
done
touch lockfile
let c=c+1
while ((c--))
do
echo "ПИШУ в файл..."
echo "Записываем в файл...">>lockfile
sleep 1
done
rm lockfile
```

Рис. 2.1: Задание 1

2. Реализовали команду `man` с помощью командного файла. Изучили 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 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`.

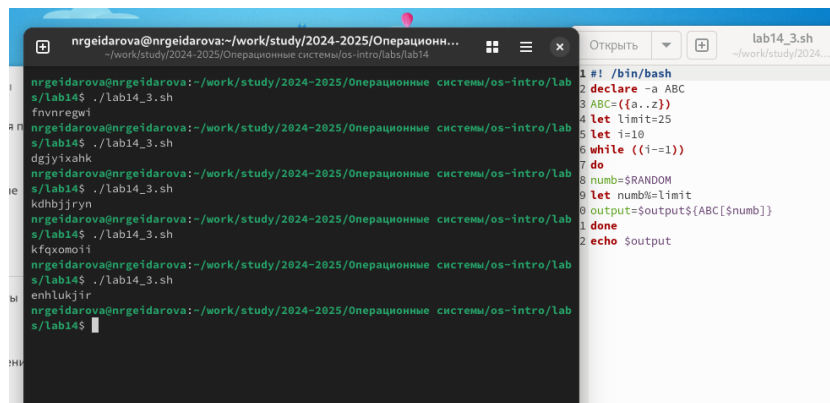


```
ESC[4mPwDESC[24m(1)
ESC[4mPwDESC[24m(1)
User Commands
ESC[1mNAMEESC[0m
    pwd - print name of current/working directory
ESC[1mSYNOPSISESC[0m
    ESC[1mpwd ESC[22m[ESC[4mOPTIONESC[24m]...
ESC[1mDESCRIPTIONESC[0m
    Print the full filename of the current working directory.
    ESC[1m-LESC[22m, ESC[1m--LogicalESC[0m
        use PWD from environment, even if it contains symlinks
    ESC[1m-PESC[22m, ESC[1m--physicalESC[0m
        avoid all symlinks
    ESC[1m--helpESC[22mdisplay this help and exit
    ESC[1m--versionESC[0m
        output version information and exit
    If no option is specified, ESC[1m-PESC[22mis assumed.
    Your shell may have its own version of pwd, which usually supersedes the
    version described here. Please refer to your shell's documentation for
    details about the options it supports.
ESC[1mAUTORESC[0m
    Written by Jim Meyering.
/usr/share/man/man1/pwd.1.gz
```

```
lab14_2.sh
~/work/study/2024-2025/Операционные системы/os-intro/labs/lab14
lab14_3.sh
~/work/study/2024-2025/Операционные системы/os-intro/labs/lab14
1 #! /bin/bash
2 less /usr/share/man/man1/$1.1.gz
```

Рис. 2.2: Задание 2

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



The image shows a terminal window on the left and a code editor on the right. The terminal window has a title bar that reads "nrgeidarova@nrgeidarova:~/work/study/2024-2025/Операционные системы/os-intro/lab14". The terminal content shows a user repeatedly running a script named "lab14_3.sh" in the directory "s/lab14". The code editor on the right has a title bar that reads "lab14_3.sh" and shows the following script content:

```
1 #!/bin/bash
2 declare -a ABC
3 ABC=({a..z})
4 let limit=25
5 let i=10
6 while ((i-->1))
7 do
8     numb=$RANDOM
9     numb%=limit
10    output=${ABC[$numb]}
11 done
12 echo $output
```

Рис. 2.3: Задание 3

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 пре-

имущества 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.