Отчёт по лабораторной работе №14

Дисциплина: Операционные системы

Верниковская Екатерина Андреевна

Содержание

# 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 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 32767.

# 3 Выполнение лабораторной работы

Создаю файл для первого задания с расширением sh и делаю его исполняемым (рис. 1)

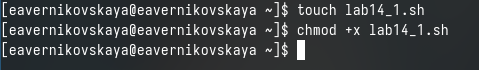


Рис. 1: Создание файла lab14\_1.sh и добавление прав на исполнение

Открываю файл lab14\_1.sh в текстовом редакторе gedit и пишу командный файл, реализующий упрощённый механизм семафоров (подробнее см. в задании №1) (рис. 2)

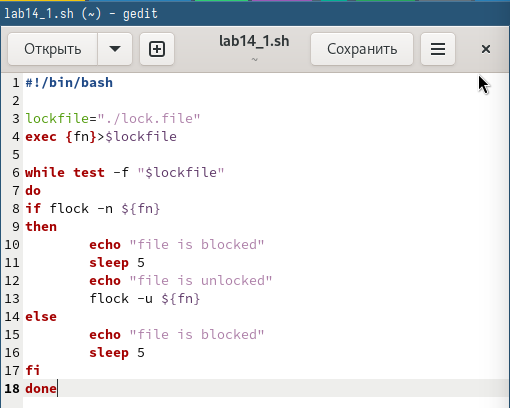


Рис. 2: Написанная программа для lab14\_1.sh

Программа для задания №1:

#!/bin/bash  
  
lockfile="./lock.file"  
exec {fn}>$lockfile  
  
while test -f "$lockfile"  
do  
if flock -n ${fn}  
then  
 echo "file is blocked"  
 sleep 5  
 echo "file is unlocked"  
 flock -u ${fn}  
else  
 echo "file is blocked"  
 sleep 5  
fi  
done

Далее запускаю файл с помощью bash и проверяю работу командного файла (рис. 3)

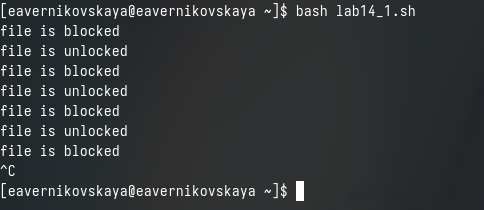


Рис. 3: Проверка работы командного файла lab14\_1.sh

Изучаю содержимое каталога /usr/share/man/man1. В нем находятся архивы текстовых файлов, содержащих справку по большинству установленных в системе программ и команд (рис. 4), (рис. 5)

Команда

Рис. 4: Команда

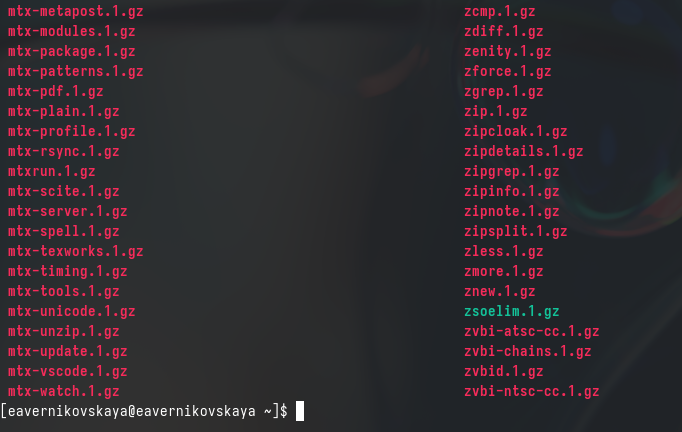


Рис. 5: Содержимое каталога /usr/share/man/man1

Создаю файл для второго задания с расширением sh и делаю его исполняемым (рис. 6)

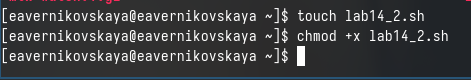


Рис. 6: Создание файла lab14\_2.sh и добавление прав на исполнение

Открываю файл lab14\_2.sh в текстовом редакторе gedit и пишу командный файл, который будет реализовывать команду man (рис. 7)

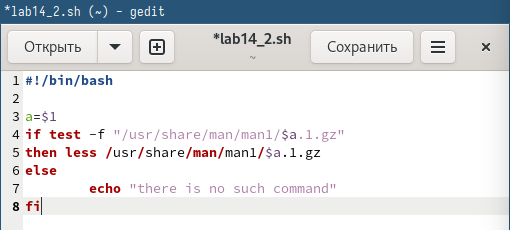


Рис. 7: Написанная программа для lab14\_2.sh

Программа для задания №2:

#!/bin/bash  
  
a=$1  
if test -f "/usr/share/man/man1/$a.1.gz"  
then less /usr/share/man/man1/$a.1.gz  
else  
 echo "there is no such command"  
fi

Далее запускаю файл с помощью bash и проверяю его работу (рис. 8), (рис. 9), (рис. 10), (рис. 11)

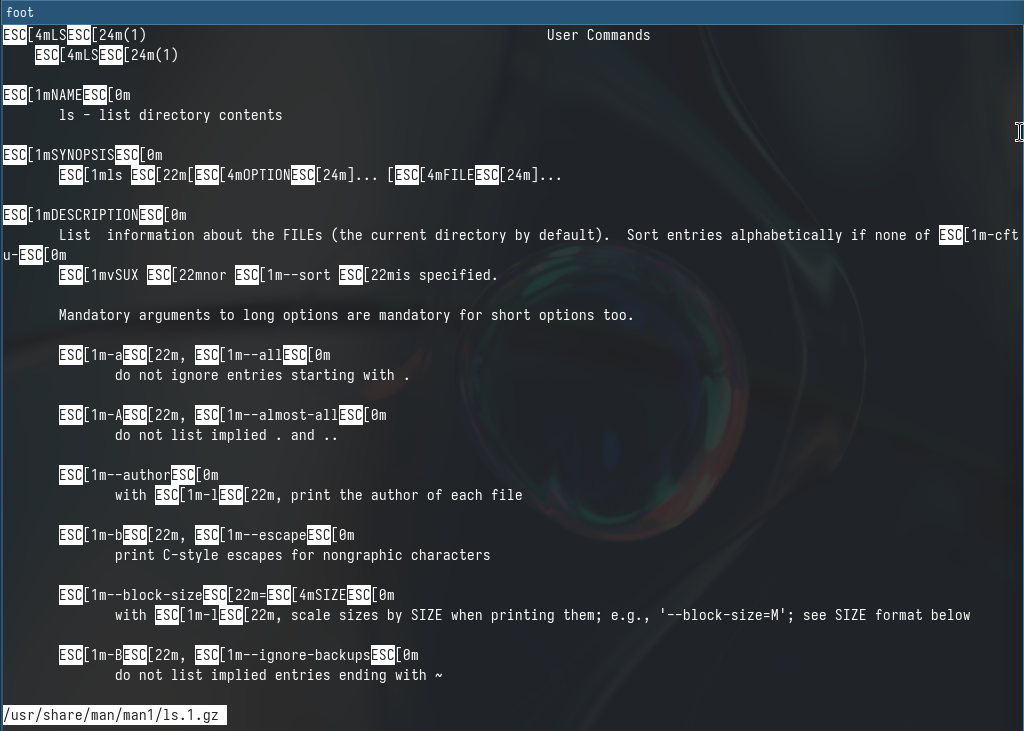


Рис. 8: man команды ls (работа командного файла lab14\_2.sh)

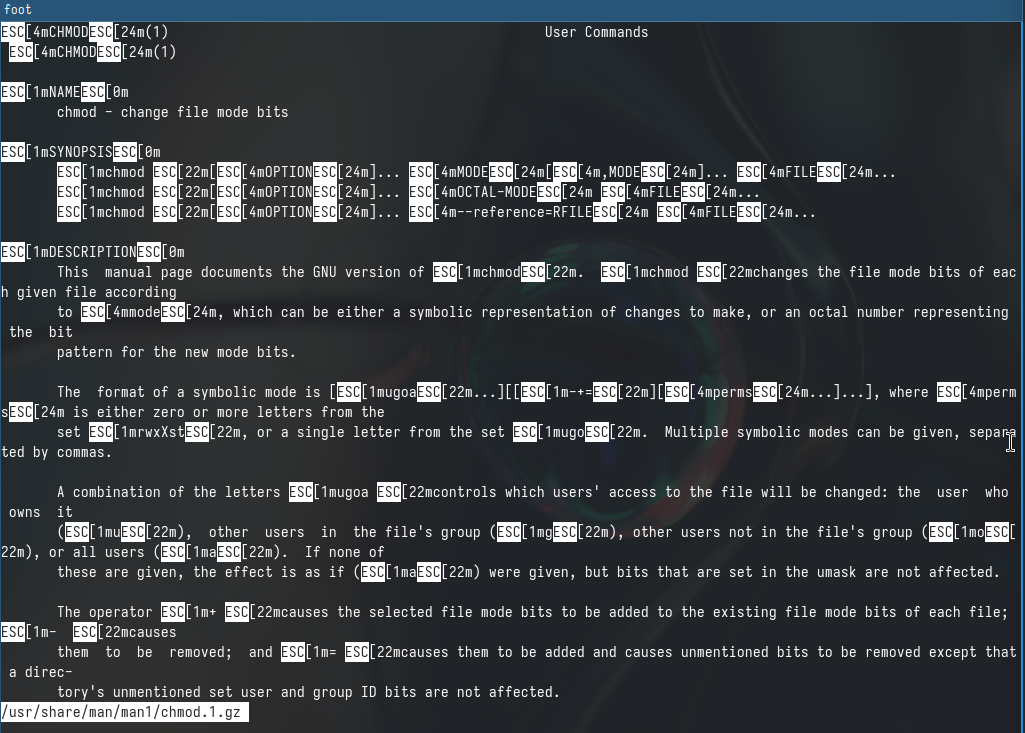


Рис. 9: man команды chmod (работа командного файла lab14\_2.sh)

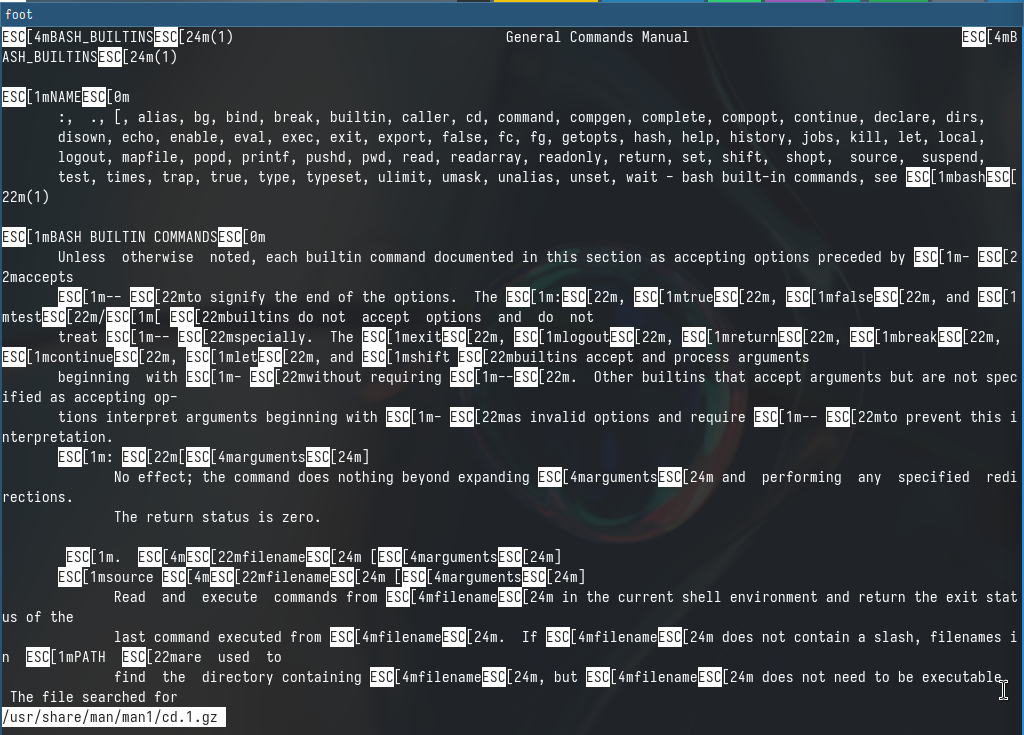


Рис. 10: man команды cd (работа командного файла lab14\_2.sh)

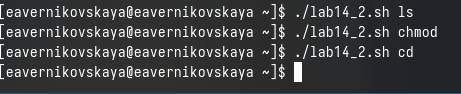


Рис. 11: Проверка работы командного файла lab14\_2.sh

Создаю файл для третьего задания с расширением sh и делаю его исполняемым (рис. 12)

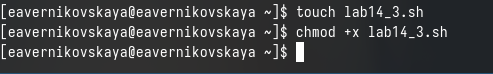


Рис. 12: Создание файла lab13\_3.sh и добавление прав на исполнение

Открываю файл lab14\_3.sh в текстовом редакторе gedit и пишу командный файл, генерирующий случайную последовательность букв латинского алфавита (рис. 13)

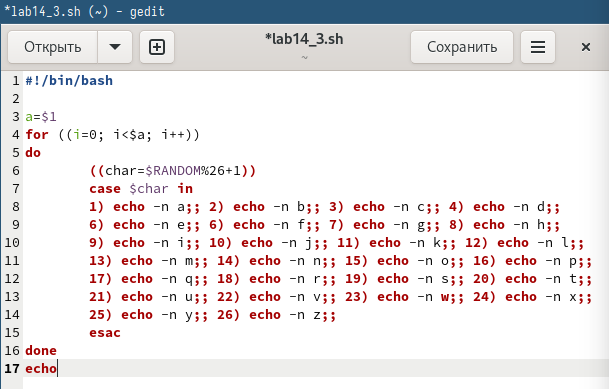


Рис. 13: Написанная программа для lab14\_3.sh

Программа для задания №3:

#!/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;;  
 6) 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

Далее запускаю файл с помощью bash и проверяю его работу (рис. 14)

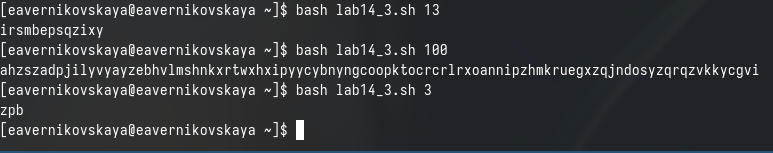


Рис. 14: Проверка работы командного файла lab14\_3.sh

# 4 Ответы на контрольные вопросы

1. Найдите синтаксическую ошибку в следующей строке:

while [$1 != "exit"]

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” ]

1. Как объединить (конкатенация) несколько строк в одну?

Чтобы объединить несколько строк в одну, можно воспользоваться несколькими способами: - Первый: VAR1=“Hello,” VAR2=” World” VAR3=“VAR2” echo “VAR1”. Результат: Hello, World

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

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.

1. Какой результат даст вычисление выражения $((10/3))?

Результатом данного выражения $((10/3)) будет 3, потому что это целочисленное деление без остатка.

1. Укажите кратко основные отличия командной оболочки zsh от bash.

Отличия командной оболочки zsh от bash: В zsh более быстрое автодополнение для cd с помощью Тab 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 основенеполных данных В zsh поддерживается замена части пути В zsh есть возможность отображать разделенный экран, такой же как разделенный экран vim

1. Проверьте, верен ли синтаксис данной конструкции

for ((a=1; a <= LIMIT; a++))

Cинтаксис данной конструкции верен, потому что, используя двойные круглые скобки, можно не писать $ перед переменными ().

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

Преимущества и недостатки 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 языков общего назначения - Утилиты, при выполнении скрипта, запускают свои процессы, которые, в свою очередь, отражаются на быстроте выполнения этого скрипта
* Скрипты, написанные на bash, нельзя запустить на других операционных системах без дополнительных действий

# 5 Выводы

В ходе выполнения лабораторной работы мы изучили основы программирования в оболочке ОС UNIX а также научились писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

# 6 Список литературы

Не пользовалась сайтами.