Лабораторная работа №12: отчет.

Программирование в командном процессоре ОС UNIX. Расширенное программирование.

Евдокимов Максим Михайлович. Группа - НФИбд-01-20.

Содержание

# Цель работы

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

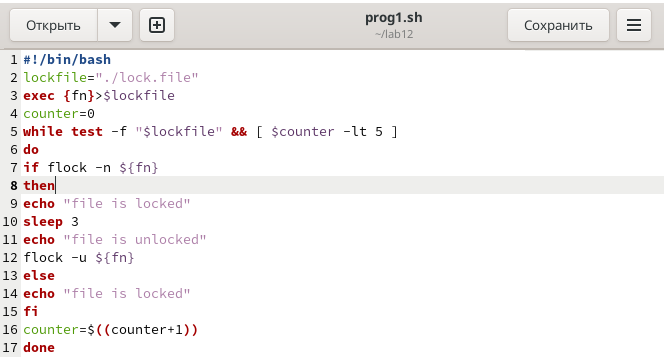
# Задание

1. Освоить работу с $RANDOM.
2. Реализация циклов с задержкой и удержанием.
3. Реализовать свой вариант команды man.

# Выполнение лабораторной работы

## 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.



Код программы 1



Результат работы 1

## Код 1

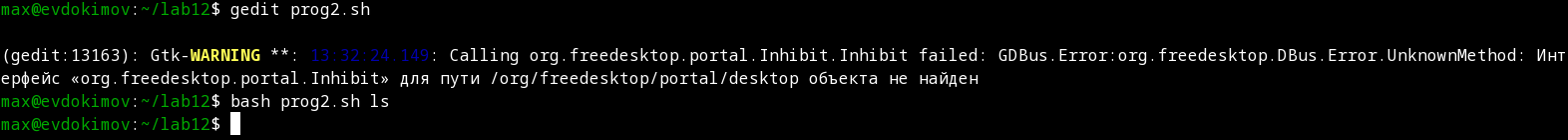
#!/bin/bash  
lockfile="./lock.file"  
exec {fn}>$lockfile  
counter=0  
while test -f "$lockfile" && [ $counter -lt 5 ]  
do  
if flock -n ${fn}  
then  
echo "file is locked"  
sleep 3  
echo "file is unlocked"  
flock -u ${fn}  
else  
echo "file is locked"  
fi  
counter=$((counter+1))  
done

## Задание 2

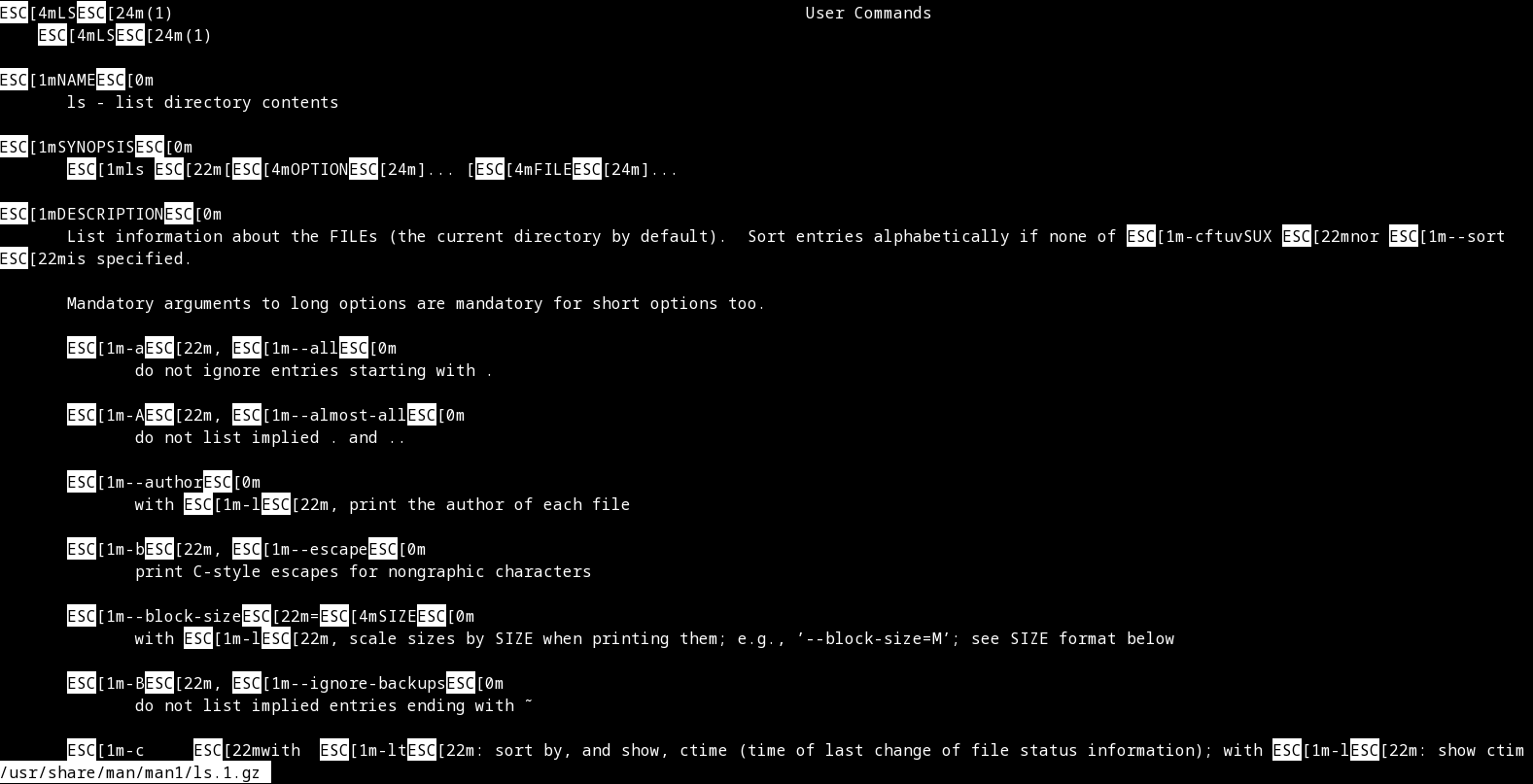
Реализовать команду man с помощью командного файла. Изучите содержимое каталога /usr/share/man/man1. В нем находятся архивы текстовых файлов, содержащих справку по большинству установленных в системе программ и команд. Каждый архив можно открыть командой less сразу же просмотрев содержимое справки. Командный 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.



Код программы 2



Вызов скрипта



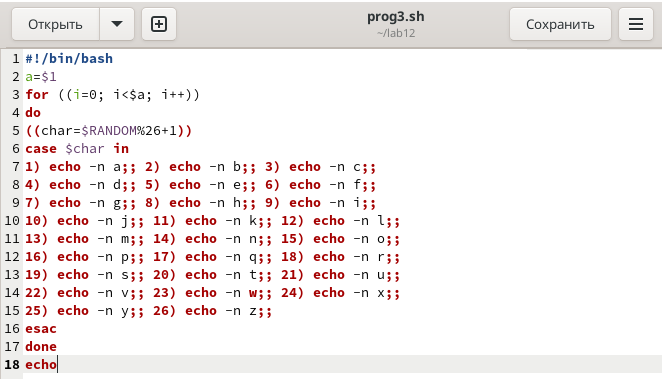
Результат работы открытие man ls

## Код 2

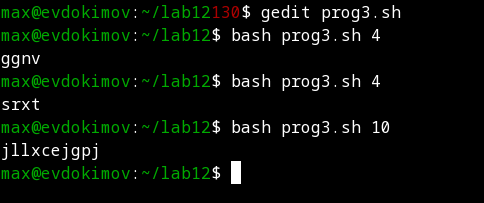
#!/bin/bash  
a=$1  
if test -f "/usr/share/man/man1/$a.1.gz"  
then less /usr/share/man/man1/$a.1.gz  
else  
echo "There is no such command"  
fi

## Задание 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.



Код программы 3



Результат работы 3

## Код 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;; 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

# 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=“VAR1VAR2” echo “$VAR3”

Результат: Hello, World

Второй: VAR1=“Hello,” VAR1+=” World” 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 основе неполных 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++))”:

“for ((a=1; a <= LIMIT; a++))” синтаксис 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 Недостат- ки 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.

# Выводы

В ходе выполнения лабораторной работы были закреплены навыки по работе с скриптами bash и изучили некоторые новые функции как RANDOM.

# Список литературы

1. [Лабораторная работа №12](https://esystem.rudn.ru/mod/resource/view.php?id=970840)