Лабораторная работа №13

Программирование в командном процессоре ОС UNIX. Расширенное программирование

Ким Михаил Алексеевич

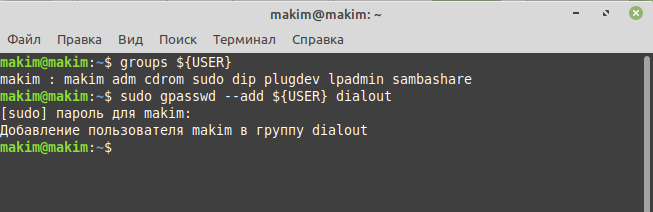
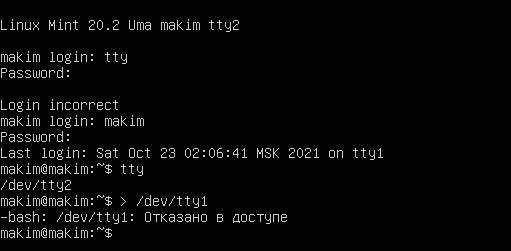
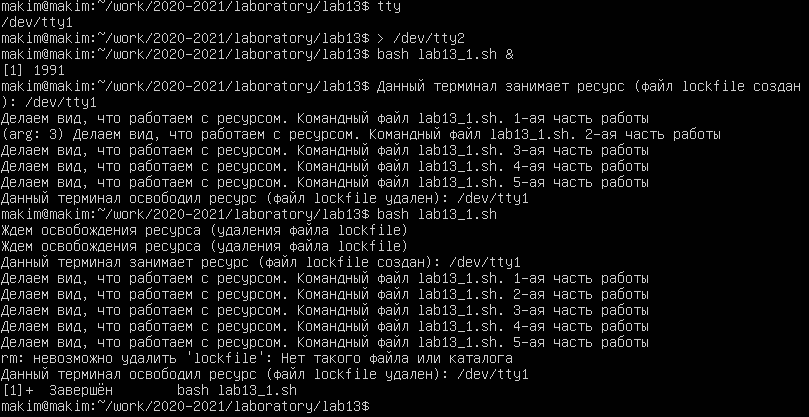
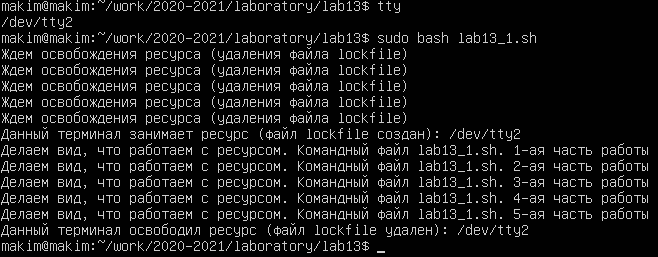
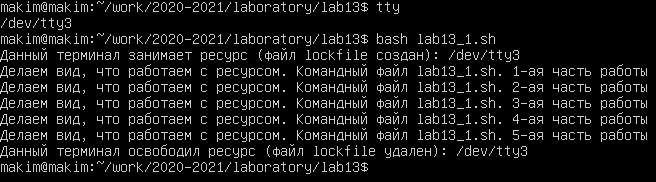
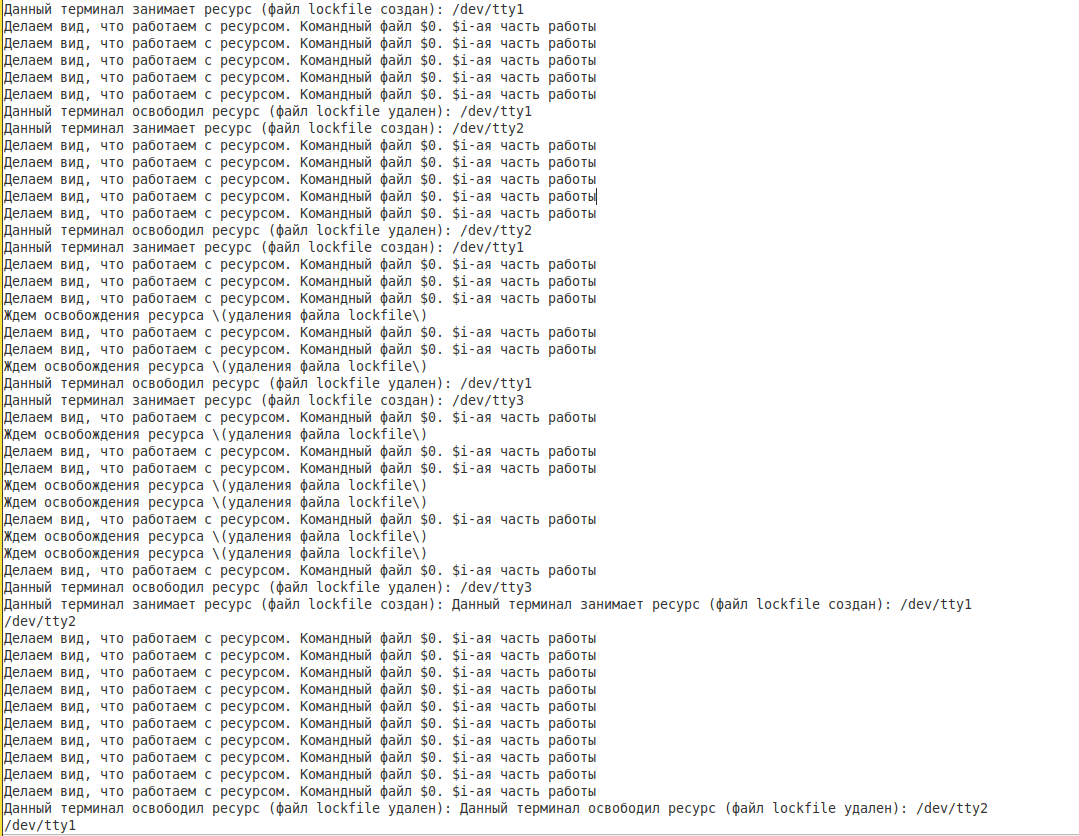
Содержание

# 1 Цель работы

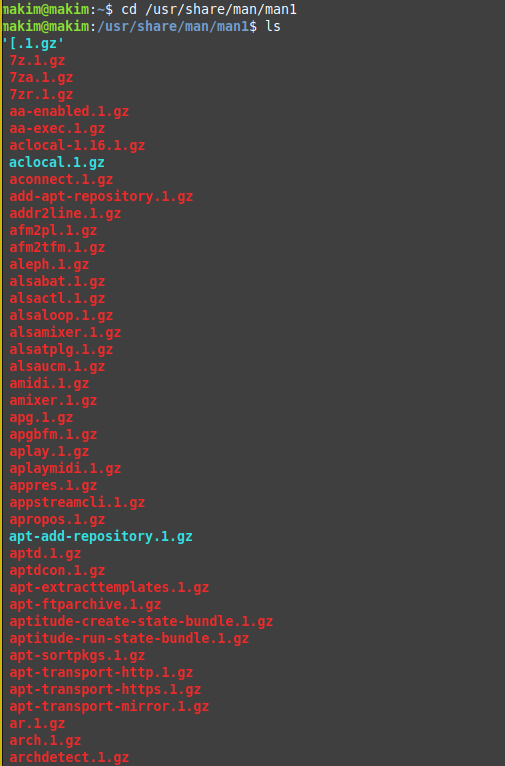
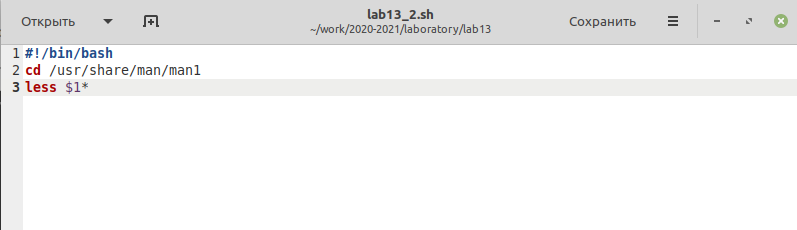
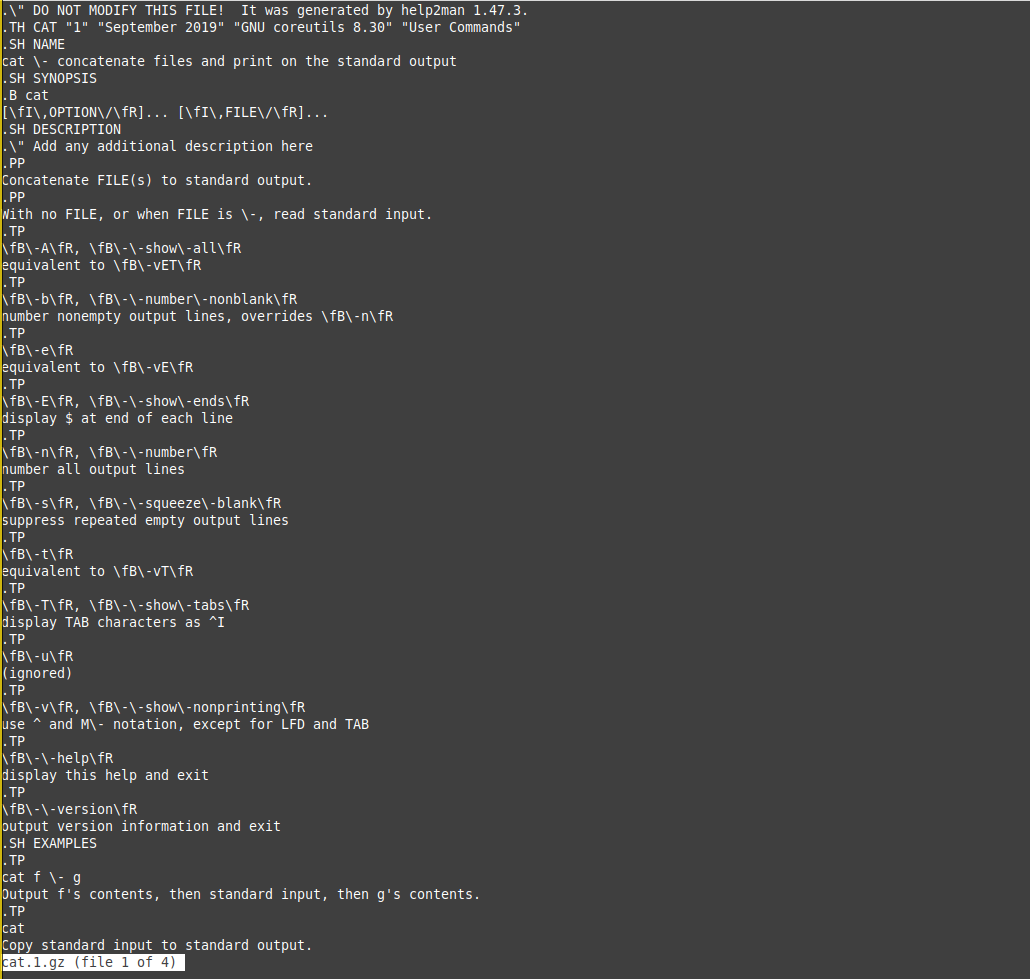
Изучить основы программирования в оболочке ОС UNIX. Научиться писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

# 2 Выполнение лабораторной работы.

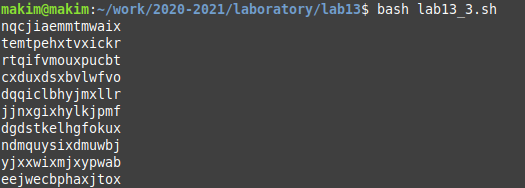
1. Написали командный файл, реализующий упрощённый механизм семафоров. Командный файл в течение некоторого времени дожидаться освобождения ресурса, выдавая об этом сообщение, а дождавшись его освобождения, использует его в течение некоторого времени, также выдавая информацию о том, что ресурс используется. К сожалению, у меня не получилось никакими методами перенеправить вывод из одного виртуального терминала на другой. (рис. 2.1 - 2.8)

* #!/bin/bash  
    
  // Пока существует файл lockfile  
  while test -f lockfile  
  do  
   echo Ждем освобождения ресурса \(удаления файла lockfile\)  
   echo 'Ждем освобождения ресурса \(удаления файла lockfile\)' >> res.txt  
   sleep 5  
  done   
    
  // Создаем файл lockfile  
  touch lockfile  
  echo -n 'Данный терминал занимает ресурс (файл lockfile создан): '  
  tty  
  echo -n 'Данный терминал занимает ресурс (файл lockfile создан): ' >> res.txt  
  tty >> res.txt  
    
  // Сообщения просто для того, чтобы показать возможность работы в это время  
  for (( i = 1; i <= 5; i++ ))  
  do  
   echo Делаем вид, что работаем с ресурсом. Командный файл $0. $i-ая часть работы  
   echo 'Делаем вид, что работаем с ресурсом. Командный файл $0. $i-ая часть работы' >> res.txt  
   sleep 3  
  done  
    
  // Удаляем файл lockfile  
  rm lockfile  
  echo -n 'Данный терминал освободил ресурс (файл lockfile удален): '  
  tty  
  echo -n 'Данный терминал освободил ресурс (файл lockfile удален): ' >> res.txt  
  tty >> res.txt
* 
* Исходный код
* 
* Проблема с виртуальными терминалами
* Проблема с виртуальными терминалами
* Проблема с виртуальными терминалами
* 
* Проблема с виртуальными терминалами
* 
* Результат
* 
* Результат
* 
* Результат
* 
* Результат

1. Реализуем команду man с помощью командного файла. Изучаем 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 получает в виде аргумента командной строки название команды и в виде результата выдаёт справку об этой команде или сообщение об отсутствии справки, если соответствующего файла нет в каталоге man1. (рис. 2.9 - 2.12)

* #!/bin/bash  
    
  // перемещаемся в каталог, где содержатся справки о командах  
  cd /usr/share/man/man1  
    
  // выводим информацию по команде  
  less $1\*
* 
* Содержимое /man1
* 
* Исходный код
* Результат
* Результат
* 
* Результат

1. Используя встроенную переменную $RANDOM, пишем командный файл, генерирующий случайную последовательность букв латинского алфавита. Учитываем, что $RANDOM выдаёт псевдослучайные числа в диапазоне от 0 до 32767. (рис. 2.12 - 2.14)

* #!/bin/bash  
    
  // создаём массив символов  
  declare -a array\_en  
    
  // заполняем массив алфавитом  
  array\_en=(a b c d e f g h i j k l m n o p q r s t u v w x y x)  
    
  // запускаем цикл до 10 итераций, что означает вывод 10 последовательностей  
  for (( i = 0; i < 10; i++))  
  do  
   // запускам цикл на случайное число итераций (до 30)  
   // это значит, что наши последовательности будут случайного размера (от 1 до 30)  
   for (( j = 1; j < 15; j++ ))  
   do  
   // выводим случайный элемент массива chars  
   // добавляем опцию -n, чтобы элементы выводились друг за другом  
   echo -n ${array\_en[$(($RANDOM % 26))]}  
   done  
    
   // переход на следующую строку  
   echo  
  done
* 
* Исходный код
* 
* Результат

# 3 Выводы

Мы изучили основы программирования в оболочке ОС UNIX. Закрепили знания, полученные в прошлых работах. Научиись писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

# 4 Термины

* Командный процессор (командная оболочка, интерпретатор команд shell) — это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера.
* POSIX (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ.
* Последовательность команд может быть помещена в текстовый файл. Такой файл называется командным.
* Флаги — это опции командной строки, обычно помеченные знаком минус; Например, для команды ls флагом может являться -F.
* Каталог, он же директория, (от англисйкого Directory) – это объект в ФС (файловой системе), необходимый для того, чтобы упросить работу с файлами.
* Домашний каталог - каталог, предназначенный для хранения собственных данных пользователя Linux. Как правило, является текущим непосредственно после регистрации пользователя в системе.
* Команда - записанный по специальным правилам текст (возможно с аргументами), представляющий собой указание на выполнение какой-либо функций (или действий) в операционной системе.