Лабораторная работа №13

Дисциплина: Операционные системы

Колчева Юлия вячеславовна

Содержание

# Цель работы

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

# Выполнение лабораторной работы

Сначала я написала 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). Для данной задачи я создала файл san.sh и написала соответствующий скрипт. (рис. 2) (рис. 1)

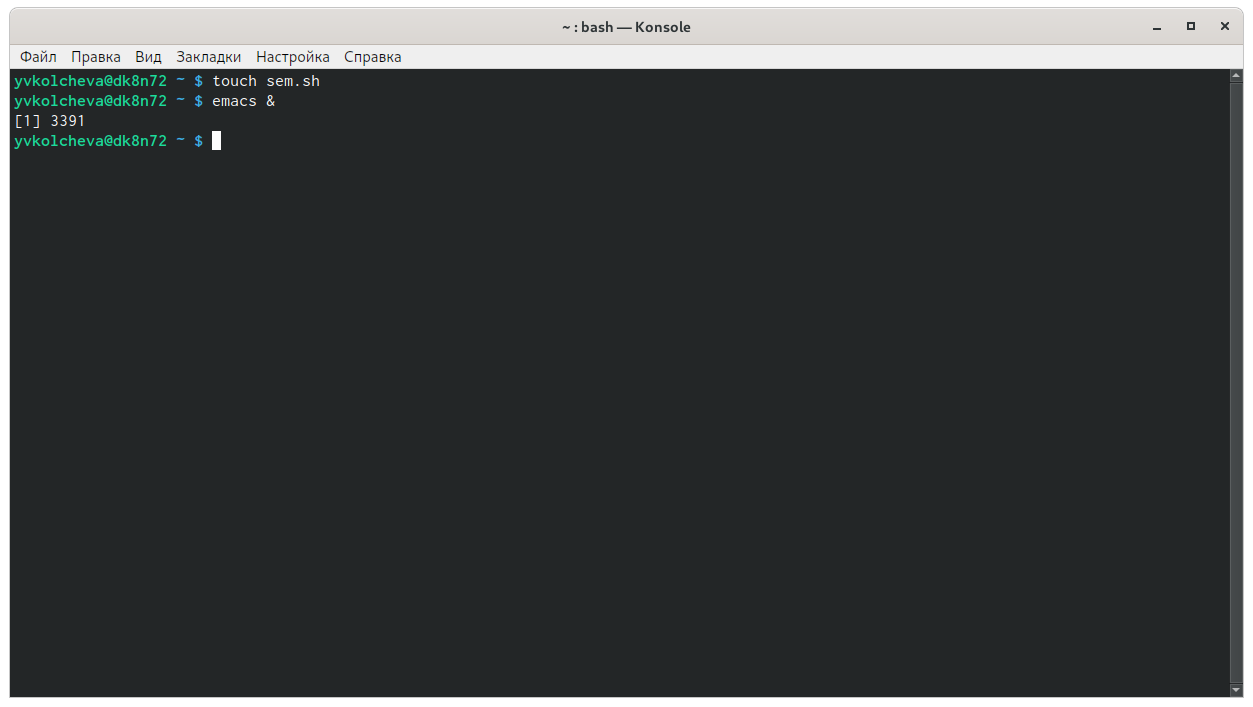


Figure 1: Создание и запуск

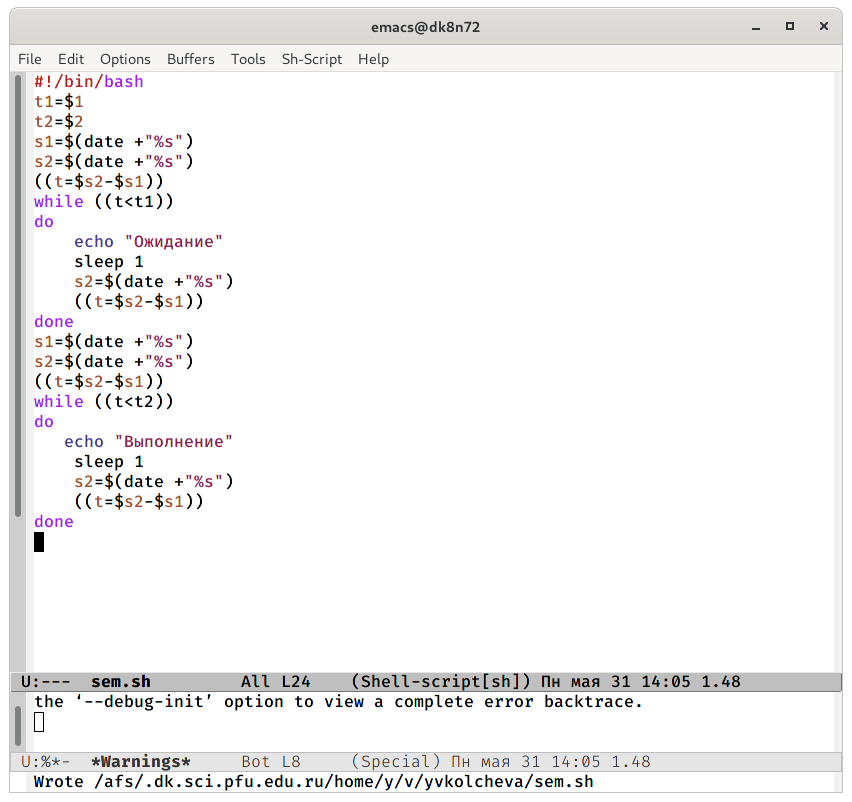


Figure 2: Скрипт1

Далее я проверила работу написанного скрипта (./prog1.sh 2 6), добавив право на исполнение файла (chmod +x \*.sh). Скрипт работает корректно. (рис. 3)

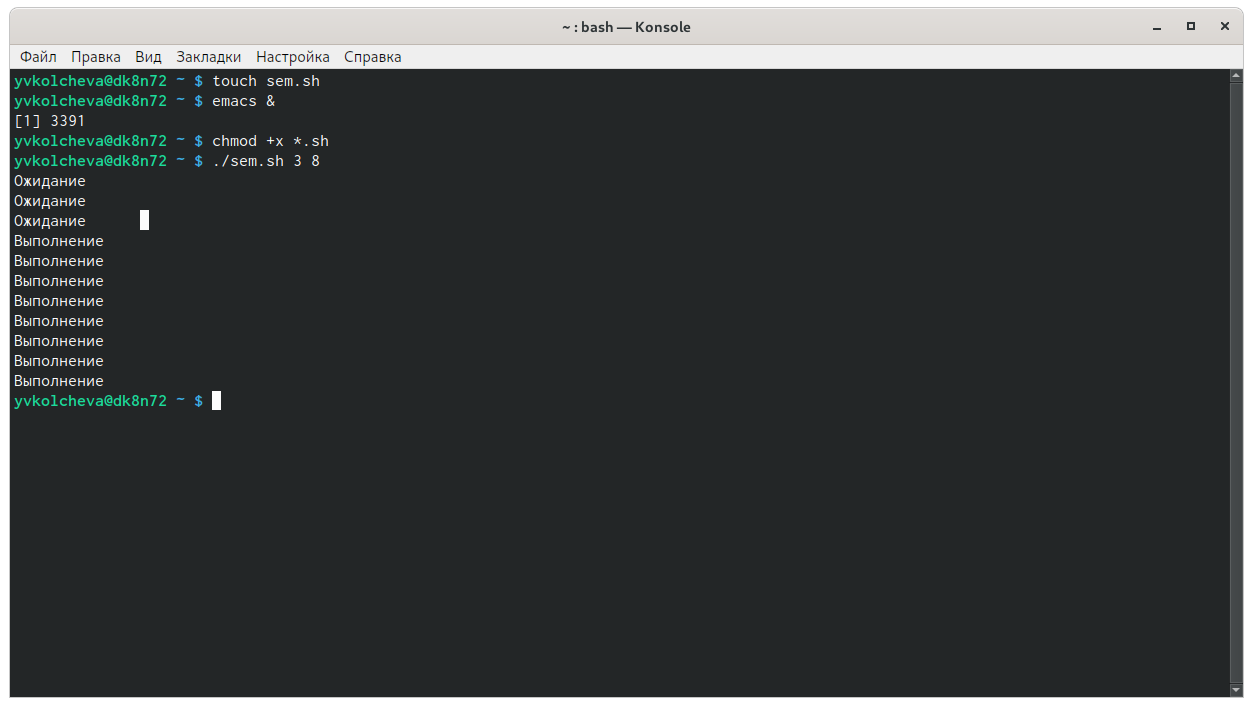


Figure 3: Проверка первого скрипта

После этого я изменила скрипт так, чтобы его можно было выполнять в нескольких терминалах и проверила его работу (./san.sh 2 5 Выполнение > /dev/tty2 &). Но ни одна команда не работала, так как мне “Отказано в доступе”. При этом скрипт работает корректно (команда «./prog1.sh 3 8»). (рис. 4) (рис. 5) (рис. 6)

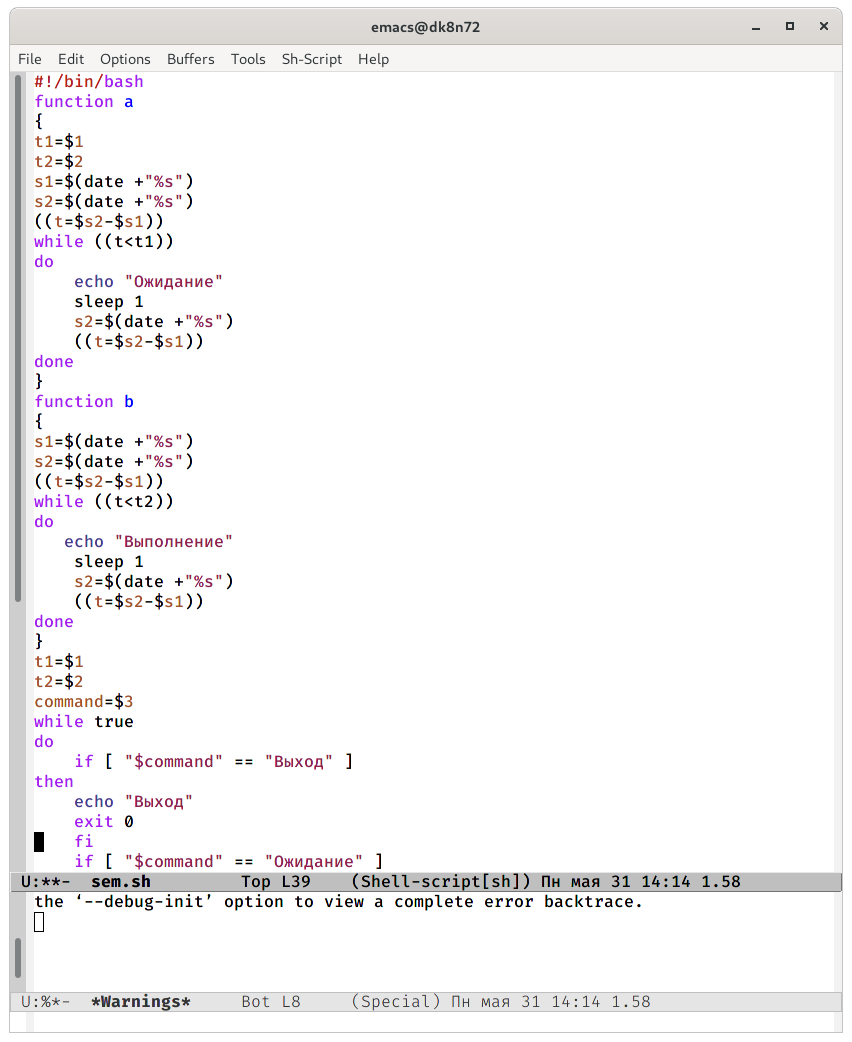


Figure 4: Первая часть скрипта

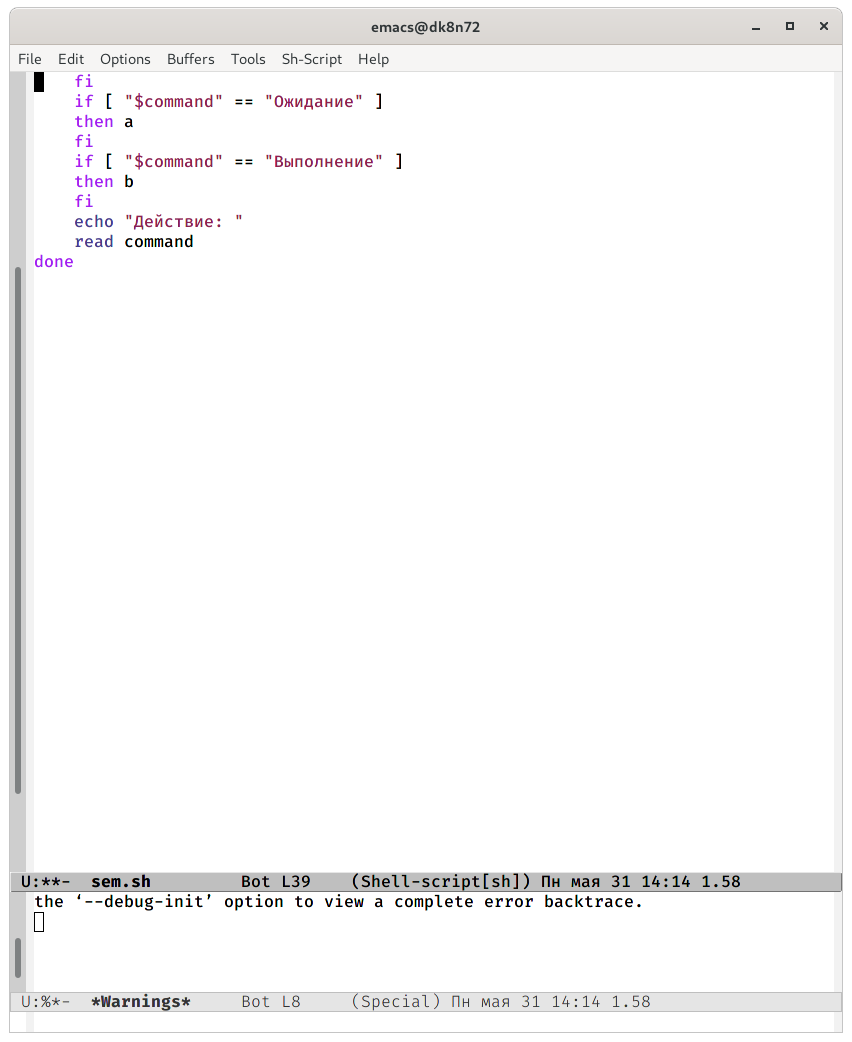


Figure 5: Вторая часть скрипта

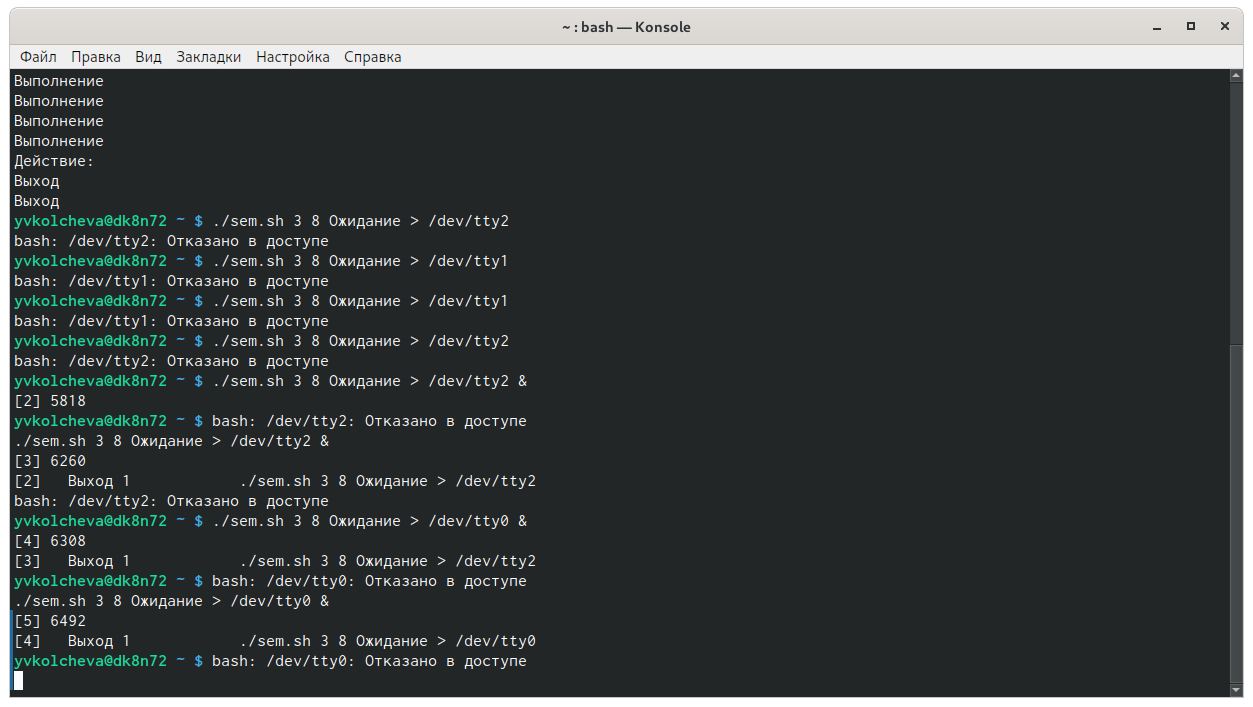
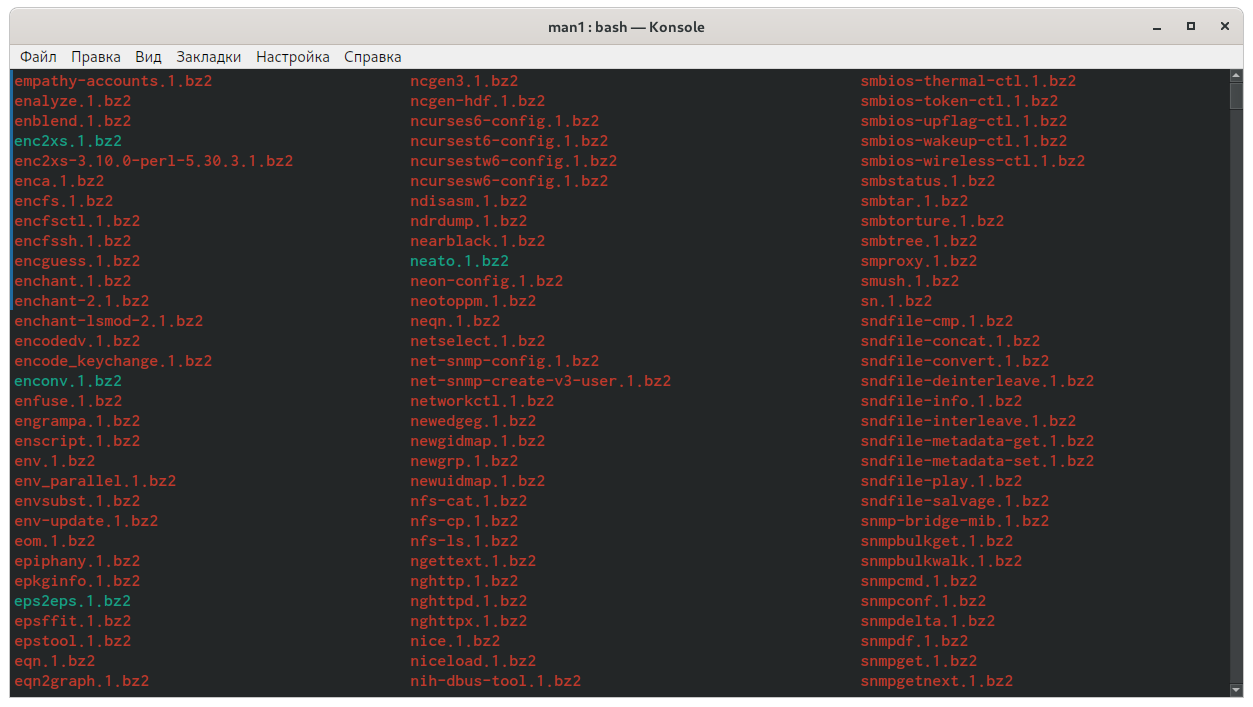


Figure 6: Выполнение

Реализовала команду man с помощью командного файла. Изучила 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. (рис. **¿fig:007?**)



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. (рис. 7)

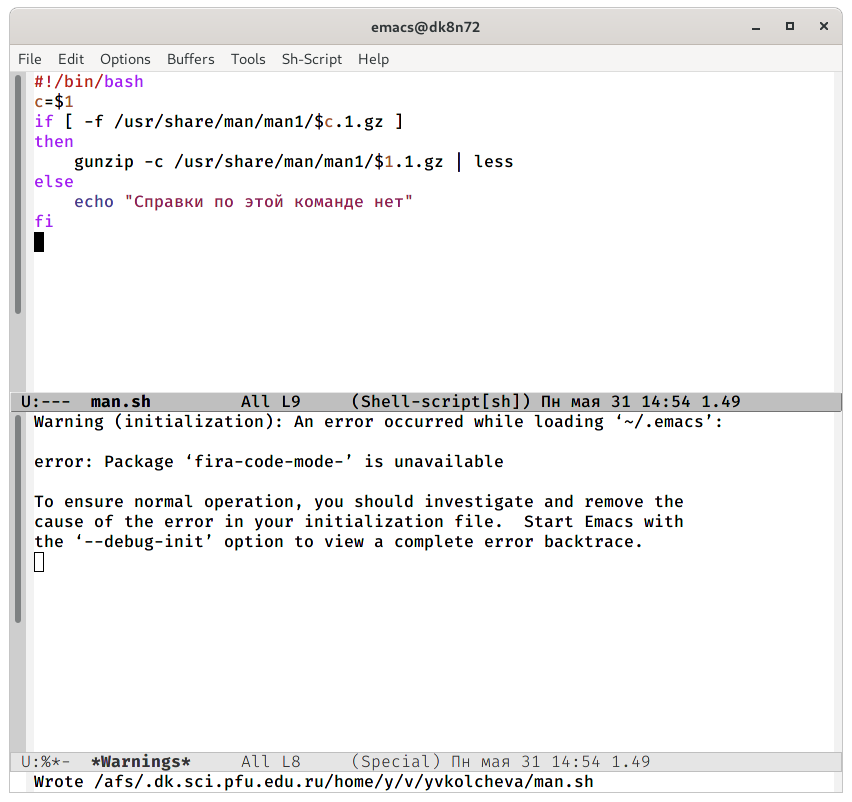


Figure 7: Скрипт3

Далее я проверила работу написанного скрипта (команды «./prog2.sh ls», «./prog2.sh mkdir»), предварительно добавив право на исполнение файла (команда «chmod +x \*.sh»). Cкрипт сработал и вывел, что по данным командам справок нет. (рис. 8)

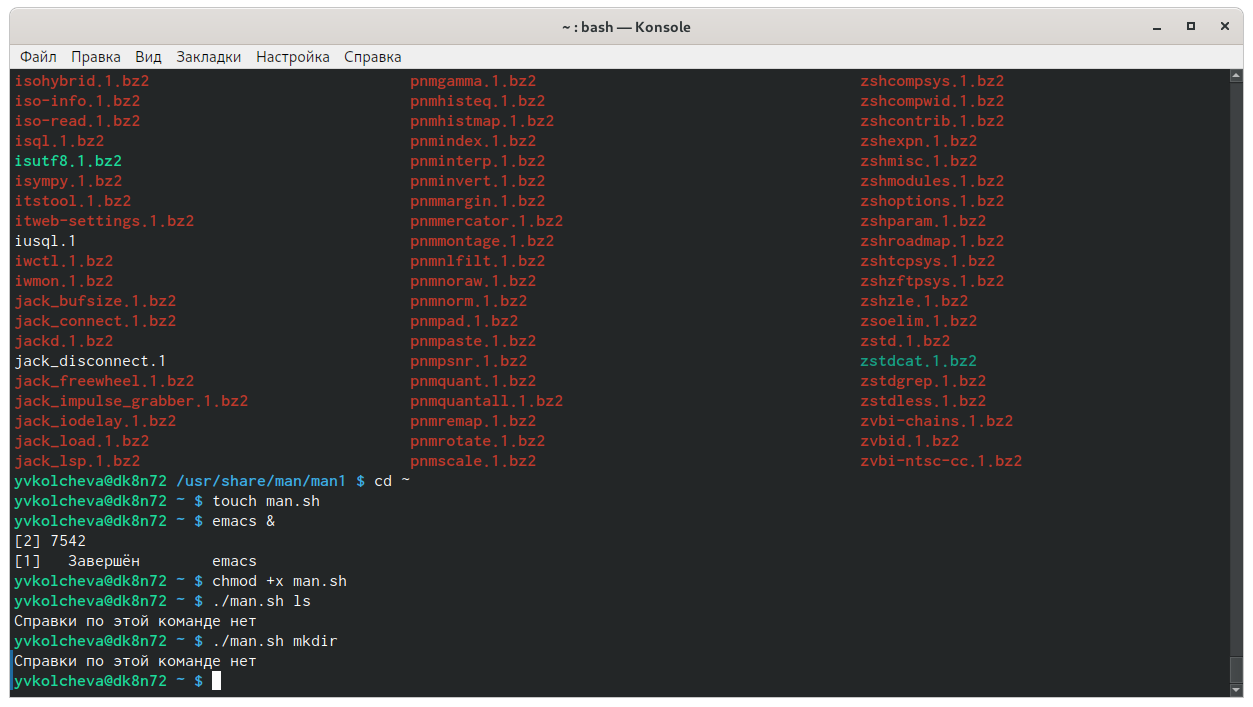


Figure 8: Проверка работы

1. Используя встроенную переменную $RANDOM, написала командный файл, генерирующий случайную последовательность букв латинского алфавита.

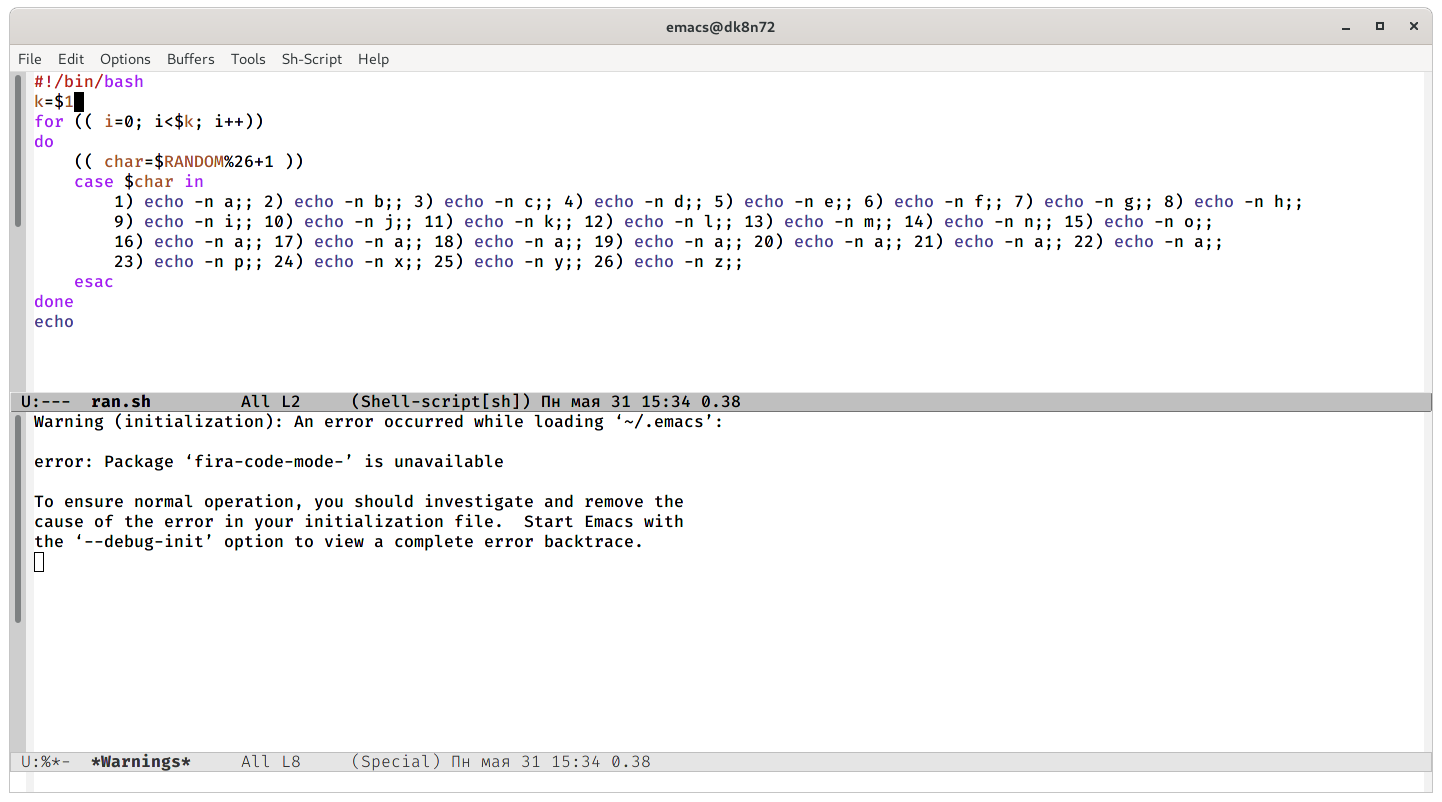


Figure 9: Последний скрипт

Далее я проверила работу написанного скрипта (./ran.sh N где N - это номер), предварительно добавив право на исполнение файла (команда «chmod +x \*.sh»). Скрипт работает корректно.(рис. 9)

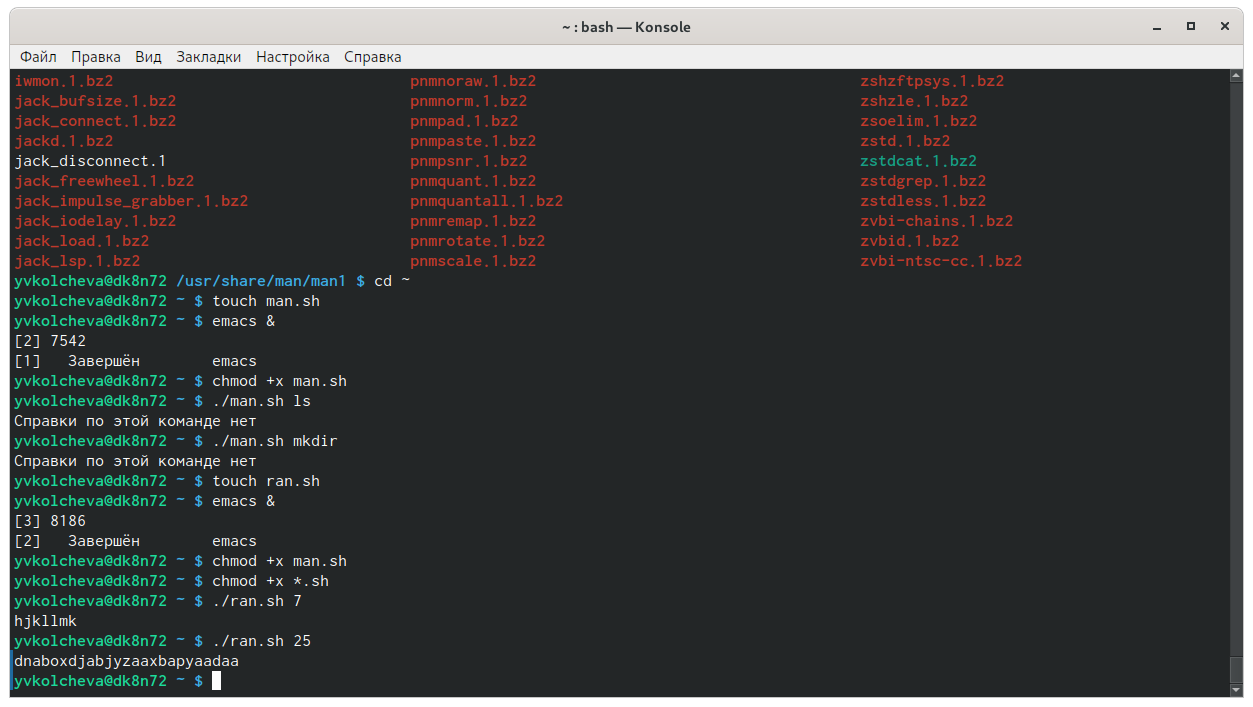


Figure 10: Проверка работы4

# Выводы

Я зучила основы программирования в оболочке ОС UNIХ и научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

# Контрольные вопросы

1)Командный процессор (командная оболочка, интерпретатор команд shell) −это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера. В операционных системах типа UNIX/Linuxнаиболее часто используются следующие реализации командных оболочек: оболочка Борна (Bourneshellили sh) −стандартная командная оболочка UNIX/Linux, содержащая базовый, но при этом полный набор функций; С-оболочка (или csh) −надстройка на оболочкой Борна, использующая Сподобный синтаксис команд с возможностью сохранения истории выполнения команд; оболочка Корна (или ksh) −напоминает оболочку С, но операторы управления программой совместимы с операторами оболочки Борна; BASH−сокращение от BourneAgainShell(опять оболочка Борна), в основе своей совмещает свойства оболочек С и Корна (разработка компании FreeSoftwareFoundation).2)POSIX(PortableOperatingSystemInterfaceforComputerEnvironments) − набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ. Стандарты POSIXразработаны комитетом IEEE(InstituteofElectricalandElectronicsEngineers) для обеспечения совместимости различных UNIX/Linuxподобных операционных систем и переносимости прикладных программ на уровне исходного кода. POSIX-совместимые оболочки разработаны на базе оболочки Корна. 3)Командный процессор bashобеспечивает возможность использования переменных типа строка символов. Имена переменных могут быть выбраны пользователем. Пользователь имеет возможность присвоить переменной значение некоторой строки символов. Например, команда «mark=/usr/andy/bin»присваивает значение строки символов /usr/andy/binпеременной markтипа строка символов. Значение, присвоенное некоторой переменной, может быть впоследствии использовано. Для этого в соответствующем месте командной строки должно быть употреблено имя этой переменной, которому предшествует метасимвол {mark}»переместит файл afileиз текущего каталога в каталог с абсолютным полным именем /usr/andy/bin.Оболочка bashпозволяет работать с массивами. Для создания массива используется команда setс флагом -A. За флагом следует имя переменной, а затем список значений, разделённых пробелами. Например, «set-AstatesDelawareMichigan“NewJersey”»Далее можно сделать добавление в массив, например, states[49]=Alaska. Индексация массивов начинается с нулевого элемента. 4)Оболочка bashподдерживает встроенные арифметические функции. Команда letявляется показателем того, что последующие аргументы представляют собой выражение, подлежащее вычислению. Простейшее выражение −это единичный терм (term), обычно целочисленный.Команда letберет два операнда и присваивает их переменной.Команда readпозволяет читать значения переменных со стандартного ввода: «echo “Please enter Month and Day of Birth ?”»«readmondaytrash»В переменные monи dayбудут считаны соответствующие значения, введённые с клавиатуры, а переменная trashнужна для того, чтобы отобрать всю избыточно введённую информацию и игнорировать её. 5)В языке программирования bashможно применять такие арифметические операции как сложение (+), вычитание (-), умножение (*), целочисленное деление (/) и целочисленный остаток от деления (%). 6)В (( ))можно записывать условия оболочки bash, а также внутри двойных скобок можно вычислять арифметические выражения и возвращать результат. 7)Стандартные переменные: PATH: значением данной переменной является список каталогов, в которых командный процессор осуществляет поиск программы или команды, указанной в командной строке, в том случае, если указанное имя программы или команды не содержит ни одного символа /. Если имя команды содержит хотя бы один символ /, то последовательность поиска, предписываемая значением переменной PATH, нарушается. В этом случае в зависимости от того, является имя команды абсолютным или относительным, поиск начинается соответственно от корневогоили текущего каталога. PS1 и PS2:эти переменные предназначены для отображения промптера командного процессора. PS1 − это промптер командного процессора, по умолчанию его значение равно символу $ или #. Если какая-то интерактивная программа, запущенная командным процессором, требует ввода, то используется промптер PS2. Он по умолчанию имеет значение символа >. HOME: имя домашнего каталога пользователя. Если команда cdвводится без аргументов, то происходит переход в каталог,указанный в этой переменной. IFS:последовательность символов, являющихся разделителями в командной строке, например, пробел, табуляция и перевод строки (newline). MAIL:командный процессор каждый раз перед выводом на экран промптера проверяет содержимое файла, имя которого указано в этой переменной, и если содержимое этого файла изменилось с момента последнего ввода из него, то перед тем как вывести на терминал промптер, командный процессор выводит на терминал сообщение Youhavemail(у Вас есть почта). TERM: тип используемого терминала. LOGNAME: содержит регистрационное имя пользователя, которое устанавливается автоматически при входе в систему. 8)Такие символы, как ’ < >*  ? | " &, являются метасимволами и имеют для командного процессора специальный смысл. 9)Снятие специального смысла с метасимвола называется экранированием метасимвола. Экранирование может быть осуществлено с помощью предшествующего метасимволу символа , который, в свою очередь, является метасимволом. Для экранирования группы метасимволов нужно заключить её в одинарные кавычки. Строка, заключённая в двойные кавычки, экранирует все метасимволы, кроме $, ’ , , ". Например, –echo\* выведет на экран символ *, –echoab’*|*’cdвыведет на экран строку ab*|*cd. 10)Последовательность команд может быть помещена в текстовый файл. Такой файл называется командным. Далее этот файл можно выполнить по команде: «bashкомандный\_файл [аргументы]»Чтобы не вводить каждый раз последовательности символов bash, необходимо изменить код защиты этого командного файла, обеспечив доступ к этому файлу по выполнению. Это может быть сделано с помощью команды «chmod+xимя\_файла»Теперь можно вызывать свой командный файл на выполнение, просто вводя его имя с терминала так, как будтоон является выполняемой программой. Командный процессор распознает, что в Вашем файле на самом деле хранится не выполняемая программа, а программа, написанная на языке программирования оболочки, и осуществит её интерпретацию. 11)Группу команд можно объединить в функцию. Для этого существует ключевое слово function, после которого следует имя функции и список команд, заключённых в фигурные скобки. Удалить функцию можно с помощью команды unsetcфлагом -f. 12)Чтобы выяснить, является ли файл каталогом или обычным файлом, необходимо воспользоваться командами «test-f путь до файла»(для проверки, является ли обычным файлом)и «test -d[путь до файла]»(для проверки, является ли каталогом). 13)Команду«set»можноиспользовать для вывода списка переменных окружения. В системах Ubuntuи Debianкоманда«set»также выведет список функций командной оболочки после списка переменных командной оболочки. Поэтому для ознакомления со всеми элементами списка переменных окружения при работе с данными системами рекомендуется использовать команду«set| more».Команда «typeset» предназначена для наложения ограничений на переменные.Команду«unset»следует использовать для удаления переменной из окружения командной оболочки. 14)При вызове командного файла на выполнение параметры ему могут быть переданы точно таким же образом, как и выполняемой программе. С точки зрения командного файла эти параметры являются позиционными. Символ $ является метасимволом командного процессора. Он используется, в частности, для ссылки на параметры, точнее, для получения их значений в командном файле. В командный файл можно передать до девяти параметров. При использовании где-либо в командном файле комбинации символов $i, где 0 < i< 10, вместо неё будет осуществлена подстановка значения параметра с порядковым номером i, т.е. аргумента командного файла с порядковым номером i. Использование комбинации символов $0 приводит к подстановке вместо неё имени данного командного файла. 15)Специальные переменные: $* −отображается вся командная строка или параметры оболочки; $? −код завершения последней выполненной команды; $$ −уникальный идентификатор процесса, в рамках которого выполняется командный процессор; $! −номер процесса, в рамках которого выполняется последняя вызванная на выполнение в командном режиме команда; $-−значение флагов командного процессора; ${#*} −возвращает целое число −количествослов, которые были результатом $*; ${#name} −возвращает целое значение длины строки в переменной name; ${name[n]} −обращение к n-му элементу массива; ${name[\*]}−перечисляет все элементы массива, разделённые пробелом; ${name[@]}−то же самое, но позволяет учитывать символы пробелы в самих переменных; ${name:-value} −если значение переменной name не определено, то оно будет заменено на указанное value; ${name:value} −проверяется факт существования переменной; ${name=value} −если name не определено, то ему присваивается значение value; ${name?value} −останавливает выполнение, если имя переменной не определено, и выводит value как сообщение об ошибке; ${name+value} −это выражение работает противоположно ${name-value}. Если переменная определена, то подставляется value; ${name#pattern} −представляет значение переменной name с удалённым самым коротким левым образцом (pattern); ${#name[\*]} и ${#name[@]}−эти выражения возвращают количество элементов в массиве name

# Библиография

Лекция Кудрявцева https://esystem.rudn.ru/mod/url/view.php?id=718562

Лабораторная работа №13 https://esystem.rudn.ru/mod/resource/view.php?id=718610