Лабораторная работа №12

Программирование в командном процессоре ОС UNIX. Расширенное программирование

Старовойтов Егор Сергеевич

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. 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.

Теоретическое введение

Командный процессор (командная оболочка, интерпретатор команд shell) — это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера. В операционных системах типа UNIX/Linux наиболее часто используются следующие реализации командных оболочек:

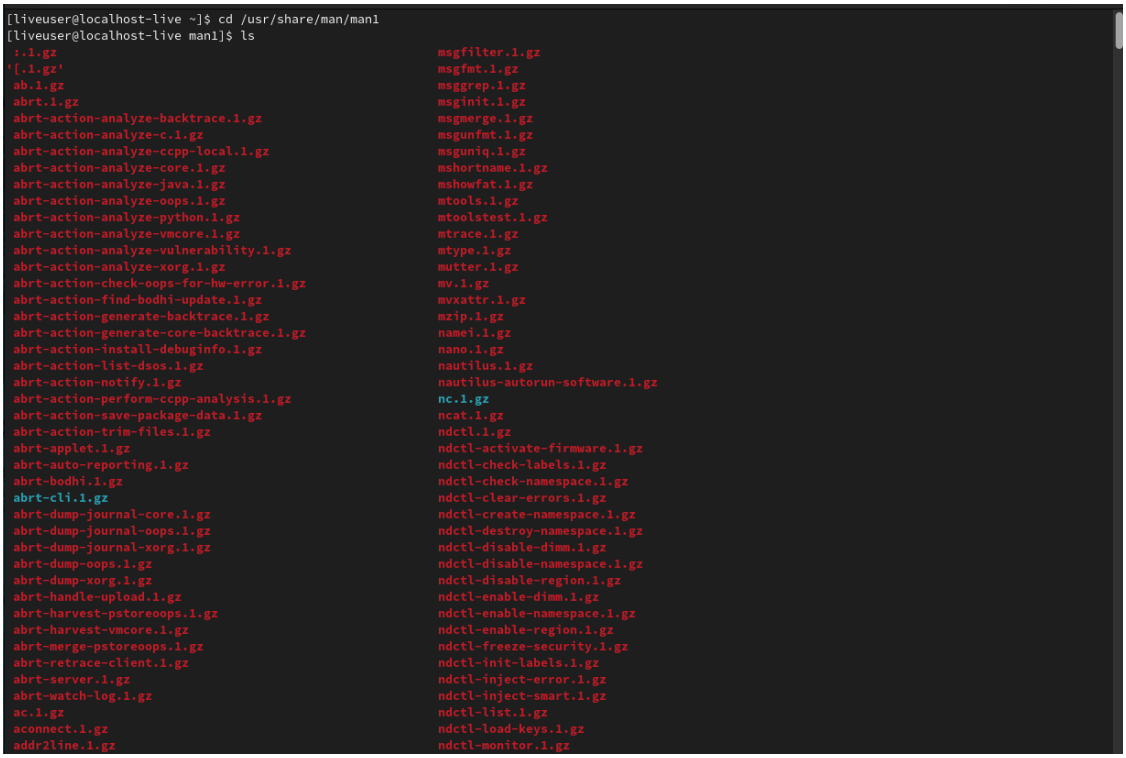
- оболочка Борна (Bourne shell или sh) — стандартная командная оболочка UNIX/Linux, содержащая базовый, но при этом полный набор функций;
- C-оболочка (или csh) — надстройка на оболочкой Борна, использующая C-подобный синтаксис команд с возможностью сохранения истории выполнения команд;
- оболочка Корна (или ksh) — напоминает оболочку C, но операторы управления программой совместимы с операторами оболочки Борна;
- BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек C и Корна (разработка компании Free Software Foundation). POSIX (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ. Стандарты POSIX

разработаны комитетом IEEE (Institute of Electrical and Electronics Engineers) для обеспечения совместимости различных UNIX/Linux-подобных операционных систем и переносимости прикладных программ на уровне исходного кода. POSIX-совместимые оболочки разработаны на базе оболочки Корна. Рассмотрим основные элементы программирования в оболочке bash. В других оболочках большинство команд будет совпадать с описанными ниже.

Ход работы

Задание 2

Я изучил содержимое каталога /usr/share/man/man1



/usr/share/man/man1

Я написал скрипт в файле prog2.sh, выполняющий второе задание, используя редактор vi.

```
#!/bin/bash

if test -f "/usr/share/man/man1/$1.1.gz"
then less /usr/share/man/man1/$1.1.gz
else echo "This command nod found"
fi
```

Код комадного файла

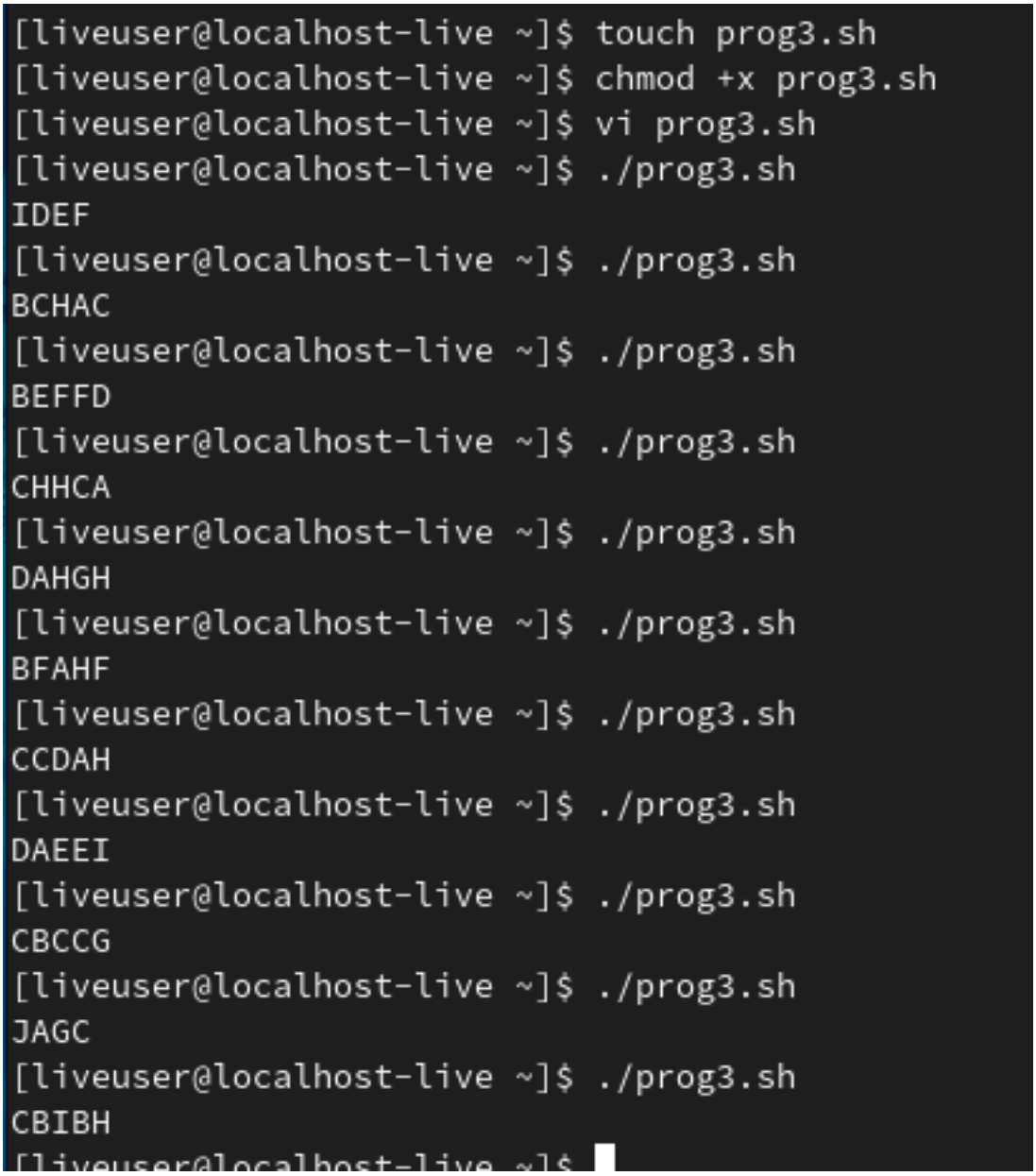
Работа программы:

Задание 3

Я написал скрипт в файле prog3.sh, выполняющий третье задание, используя редактор vi.

```
#!/bin/bash
echo $RANDOM | tr '0-9' 'A-Z'
[liveuser@localhost-live ~]$
```

Код командного файла



Работа программы

Вывод

Я изучил основы программирования в оболочке ОС UNIX. Научился писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

Контрольные вопросы

1. Найдите синтаксическую ошибку в следующей строке: `while [$1 != "exit"]`

Не хватает пробелов после [и до].

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

С помощью оператора + над строковым типом, например

```
s1 = "Te"
s1 += "xt
```

После чего s1 будет содержать “Text”.

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

Команда seq выводит последовательность целых или действительных чисел, подходящую для передачи в другие программы. Команда seq может пригодиться в различных других командах и циклах для генерации последовательности чисел.

Команду seq можно реализовать с помощью цикла while.

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

3, так как это целочисленное деление без остатка.

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

В zsh есть хеш-таблицы, числа с плавающей точкой, разделение экрана, встроенный калькулятор zcalc.

6. Проверьте, верен ли синтаксис данной конструкции `for ((a=1; a <= LIMIT; a++))`

Синтаксис верен.

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

Плюсы: - Bash распространен и универсален - С его помощью можно автоматиизровать работу в терминале - Удобное управление вводом/выводом - Возможность легкого вызова сторонних программ

Минусы: - Непереносимость кода на другие операционные системы - Другие языки могут расширять свои возможности с помощью библиотек, а bash только с помощью установки программного обеспечения.