Лабораторная работа №14

Петлин А. Д.

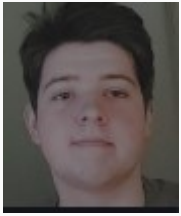
17 мая 2025

Российский университет дружбы народов, Москва, Россия

Информация

Докладчик

- Петлин Артём Дмитриевич
- студент
- группа НПИбд-02-24
- Российский университет дружбы народов
- 1132246846@pfur.ru
- https://github.com/hikrim/study_2024-2025_os-intro



Цель работы

Цель работы

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

Задание

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.

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.
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 про- грамма, 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).

Выполнение лабораторной работы

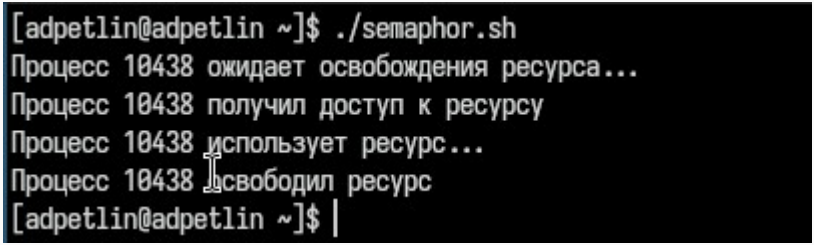
Ход работы

Пишем командный файл, реализующий упрощённый механизм семафоров.

```
#!/bin/bash
LOCK_FILE="/tmp/resource.lock"
T1=5
T2=3

echo "Процесс $$ ожидает освобождения ресурса..."
while [ -f "$LOCK_FILE" ] && [ $T1 -gt 0 ]; do
    echo "Ресурс занят, ожидание..."
    sleep 1
    ((T1--))
done

if [ $T1 -le 0 ]; then
    echo "Процесс $$: время ожидания истекло, выход"
    exit 1
fi

touch "$LOCK_FILE"
echo "Процесс $$ получил доступ к ресурсу"

echo "Процесс $$ использует ресурс..."
sleep $T2

rm -f "$LOCK_FILE"
echo "Процесс $$ освободил ресурс"
```

```
[adpetlin@adpetlin ~]$ chmod +x semaphor.sh
[adpetlin@adpetlin ~]$ ./semaphor.sh > /dev/tty2
[adpetlin@adpetlin ~]$
```

Перенаправляем вывод из первого терминала во второй.



Вывод во втором терминале.

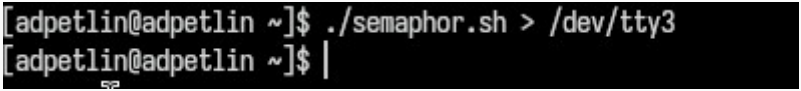
Ход работы

Дорабатываем программу так, чтобы имелась возможность взаимодействия трёх и более процессов.

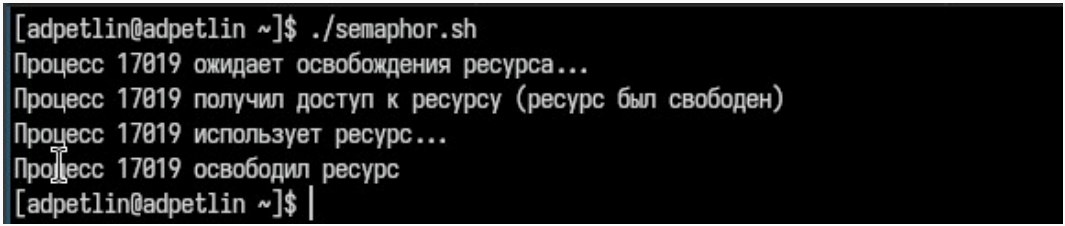
```bash
#!/bin/bash

LOCK_FILE="/tmp/resource.lock"
FIFO_DIR="/tmp/resource_fifos"
T1=5
T2=3

mkdir -p "$FIFO_DIR"

FIFO_PATH="$FIFO_DIR/$$"
mkfifo "$FIFO_PATH"

cleanup() {
    rm -f "$FIFO_PATH"
    if [ -f "$LOCK_FILE" ] && [ "$(cat "$LOCK_FILE")" = "$$" ]; then
        rm -f "$LOCK_FILE"
        echo "Процесс $$ освободил ресурс при завершении"
        next_fifo=$(ls "$FIFO_DIR" | sort -n | head -n 1)
        if [ -n "$next_fifo" ]; then
            echo "1" > "$FIFO_DIR/$next_fifo"
        fi
    fi
    exit
}

trap cleanup EXIT SIGINT SIGTERM

echo "Процесс $$ ожидает освобождения ресурса..."
```

```bash
if [ ! -f "$LOCK_FILE" ]; then
    echo "$$" > "$LOCK_FILE"
    echo "Процесс $$ получил доступ к ресурсу (ресурс был свободен)"
else
    echo "Процесс $$ встал в очередь на ресурс"
    while read -r; do
        if [ ! -f "$LOCK_FILE" ]; then
            echo "$$" > "$LOCK_FILE"
            echo "Процесс $$ получил доступ к ресурсу (получил уведомление)"
            break
        fi
    done < "$FIFO_PATH"
fi

echo "Процесс $$ использует ресурс..."
sleep $T2

rm -f "$LOCK_FILE"
echo "Процесс $$ освободил ресурс"

next_fifo=$(ls "$FIFO_DIR" | sort -n | grep -v "^$$$" | head -n 1)
if [ -n "$next_fifo" ]; then
    echo "1" > "$FIFO_DIR/$next_fifo"
fi

rm -f "$FIFO_PATH"
```

Перенаправляем вывод из первого терминала во второй.



Перенаправляем вывод из второго терминала в третий.



Вывод в третьем терминале

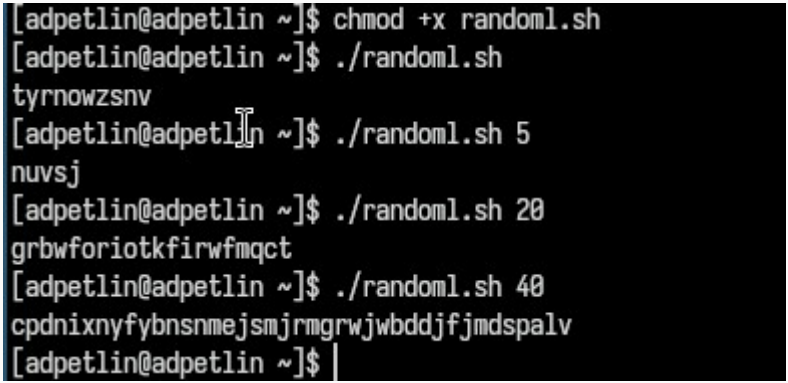
```
#!/bin/bash

MAN_DIR="/usr/share/man/man1"
COMMAND=$1
MAN_FILE="$MAN_DIR/${COMMAND}.1.gz"

if [ -f "$MAN_FILE" ]; then
    gunzip -c $MAN_FILE | less
else
    echo "Справка по команде '$COMMAND' не найдена."
fi
```

Реализовываем команду man с помощью командного файла.

```
.\" DO NOT MODIFY THIS FILE!  It was generated by help2man 1.48.5.
.TH PWD "1" "November 2024" "GNU coreutils 9.5" "User Commands"
.SH NAME
pwd \- print name of current/working directory
.SH SYNOPSIS
.B pwd
[\fI\,OPTION\/\fR]...
.SH DESCRIPTION
.\" Add any additional description here
.PP
Print the full filename of the current working directory.
.TP
\fB\-L\fR, \fB\-\-logical\fR
use PWD from environment, even if it contains symlinks
.TP
\fB\-P\fR, \fB\-\-physical\fR
avoid all symlinks
```

```
#!/bin/bash

LENGTH=${1:-10}  # Длина последовательности (по умолчанию 10)

for ((i=0; i<LENGTH; i++)); do
    RAND_NUM=$((RANDOM % 26))
    LETTER=$(printf \\$(printf '%03o' $((97 + RAND_NUM))))
    echo -n "$LETTER"
done
echo
```

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



Вывод при разных аргументах или их отсутсвии.

Выводы

Выводы

Мы изучили основы программирования в оболочке ОС UNIX. Научились писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

Список литературы

Список литературы

1. Dash, P. Getting Started with Oracle VM VirtualBox / P. Dash. – Packt Publishing Ltd, 2013. – 86 сс.
2. Colvin, H. VirtualBox: An Ultimate Guide Book on Virtualization with VirtualBox. VirtualBox / H. Colvin. – CreateSpace Independent Publishing Platform, 2015. – 70 сс.
3. Vugt, S. van. Red Hat RHCSA/RHCE 7 cert guide : Red Hat Enterprise Linux 7 (EX200 and EX300) : Certification Guide. Red Hat RHCSA/RHCE 7 cert guide / S. van Vugt. – Pearson IT Certification, 2016. – 1008 сс.
4. Робачевский, А. Операционная система UNIX / А. Робачевский, С. Немнюгин, О. Стесик. – 2-е изд. – Санкт-Петербург : БХВ-Петербург, 2010. – 656 сс.
5. Немет, Э. Unix и Linux: руководство системного администратора. Unix и Linux / Э. Немет, Г. Снайдер, Т.Р. Хейн, Б. Уэйли. – 4-е изд. – Вильямс, 2014. – 1312 сс.
6. Колисниченко, Д.Н. Самоучитель системного администратора Linux : Системный администратор / Д.Н. Колисниченко. – Санкт-Петербург : БХВ-Петербург, 2011. – 544 сс.
7. Robbins, A. Bash Pocket Reference / A. Robbins. – O'Reilly Media, 2016. – 156 сс.