



Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУК «Информатика и управление»

КАФЕДРА ИУК4 «Программное обеспечение ЭВМ, информационные технологии»

ЛАБОРАТОРНАЯ РАБОТА №8

«Создание сценария для конфигурации системы»

ДИСЦИПЛИНА: «Операционные системы»

Выполнил: студент гр. ИУК4-62Б

_____ (Калашников А.С.)
(Подпись) (Ф.И.О.)

Проверил:

_____ (Красавин Е.В.)
(Подпись) (Ф.И.О.)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:

- Оценка:

Цель: закрепление полученных навыков по настройке основных сервисов системы FreeBSD.

Задачи:

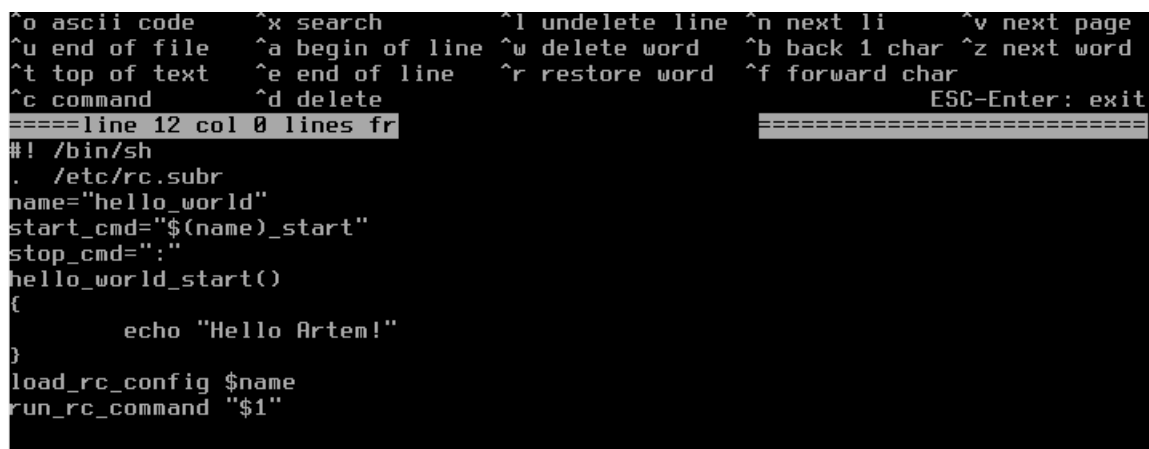
1. Сконфигурировать систему исходя из заданной вам схемы сети (сетевые интерфейсы, маршрутизация, DNS).

Задание:

Сконфигурировать систему исходя из заданной вам схемы сети (сетевые интерфейсы, маршрутизация, DNS). Продемонстрировать работу команд. Выполнить следующие шаги:

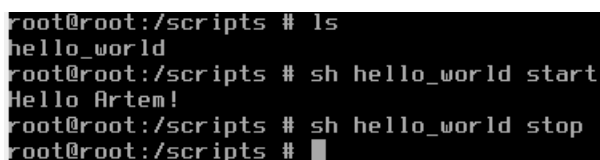
1. Ознакомиться с предложенным материалом для получения информации об управлении учетными записями в ОС FreeBSD
2. Создать простейший сценарий.
3. Настроить простейший сценарий.
4. Создать простейший демон.
5. Запустить простейший демон.
6. Создать более сложный демон.
7. Присоединить сценарий к инфраструктуре.
8. Написать сценарий исходя из заданной схемы сети.
9. Проверить работоспособность написанного скрипта.
10. Сделать скрипт более гибким. Ответить на контрольные вопросы и подготовить отчет.

Описание процесса выполнения лабораторной работы:



```
^o ascii code      ^x search          ^l undelete line   ^n next li         ^v next page
^u end of file     ^a begin of line   ^w delete word     ^b back 1 char     ^z next word
^t top of text     ^e end of line    ^r restore word    ^f forward char
^c command         ^d delete
ESC-Enter: exit
=====line 12 col 0 lines fr=====
#!/bin/sh
. /etc/rc.subr
name="hello_world"
start_cmd="$(name)_start"
stop_cmd=""
hello_world_start()
{
    echo "Hello Artem!"
}
load_rc_config $name
run_rc_command "$1"
```

Рис.1. Создание простейшего rc.d скрипта



```
root@root:/scripts # ls
hello_world
root@root:/scripts # sh hello_world start
Hello Artem!
root@root:/scripts # sh hello_world stop
root@root:/scripts #
```

Рис.2. Демонстрация работы простейшего rc.d скрипта

```

^I (escape) menu ^y search prompt ^k delete line ^p prev li ^g prev page
^o ascii code ^x search ^l undelete line ^n next li ^v next page
^u end of file ^a begin of line ^w delete word ^b back 1 char ^z next word
^t top of text ^e end of line ^r restore word ^f forward char
^c command ^d delete char ^j undelete char ESC-Enter: exit
=====line 18 col 10 lines from top 18 =====
#!/bin/sh
. /etc/rc.subr

name="chappie"

start_cmd="${name}_start"
stop_cmd=""

hello_cmd="${name}_hello"
extra_commands="hello"

chappie_start()
{
    if [ $# -gt 0 ]; then
        echo "Chappie start: $*"
    else
        echo "Start... I don't know what"
    fi
}

chappie_hello()
{
    if [ $# -gt 0 ]; then
        echo "\"$*\"? Sorry, Chappie doesn't understand you."
    else
        echo "Hello! Chappie is glad to see you."
    fi
}

load_rc_config $name
run_rc_command "$@"

```

Рис.3. Создание более гибкого rc.d скрипта

```

root@root:/scripts # sh chappie start
Start... I don't know what
root@root:/scripts # sh chappie start dancing
Chappie start: dancing
root@root:/scripts # sh chappie hello whats up
"whats up"? Sorry, Chappie doesn't understand you.
root@root:/scripts # sh chappie hello
Hello! Chappie is glad to see you.
root@root:/scripts #

```

Рис.4. Демонстрация работы более гибкого скрипта

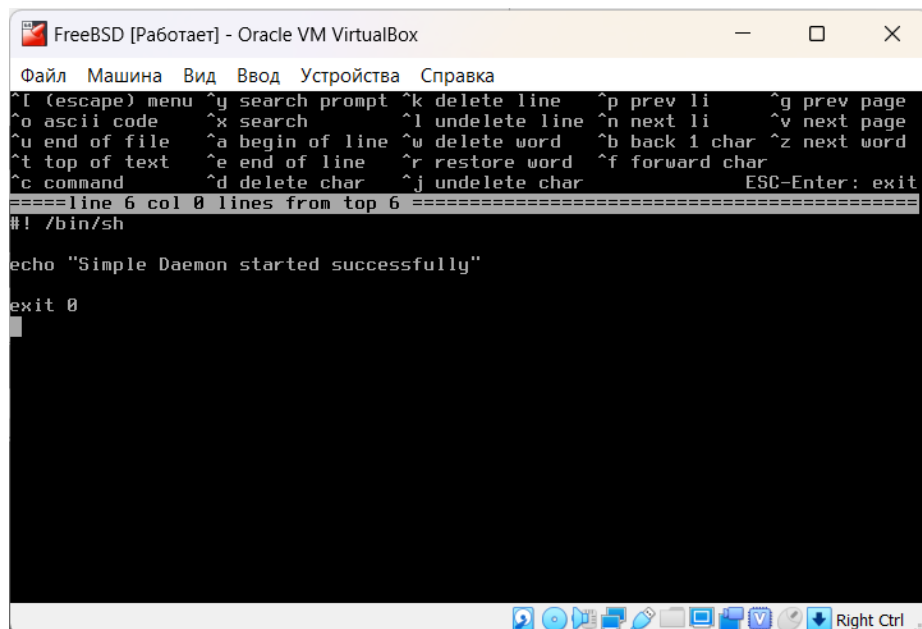


Рис.5. Создание простого демона

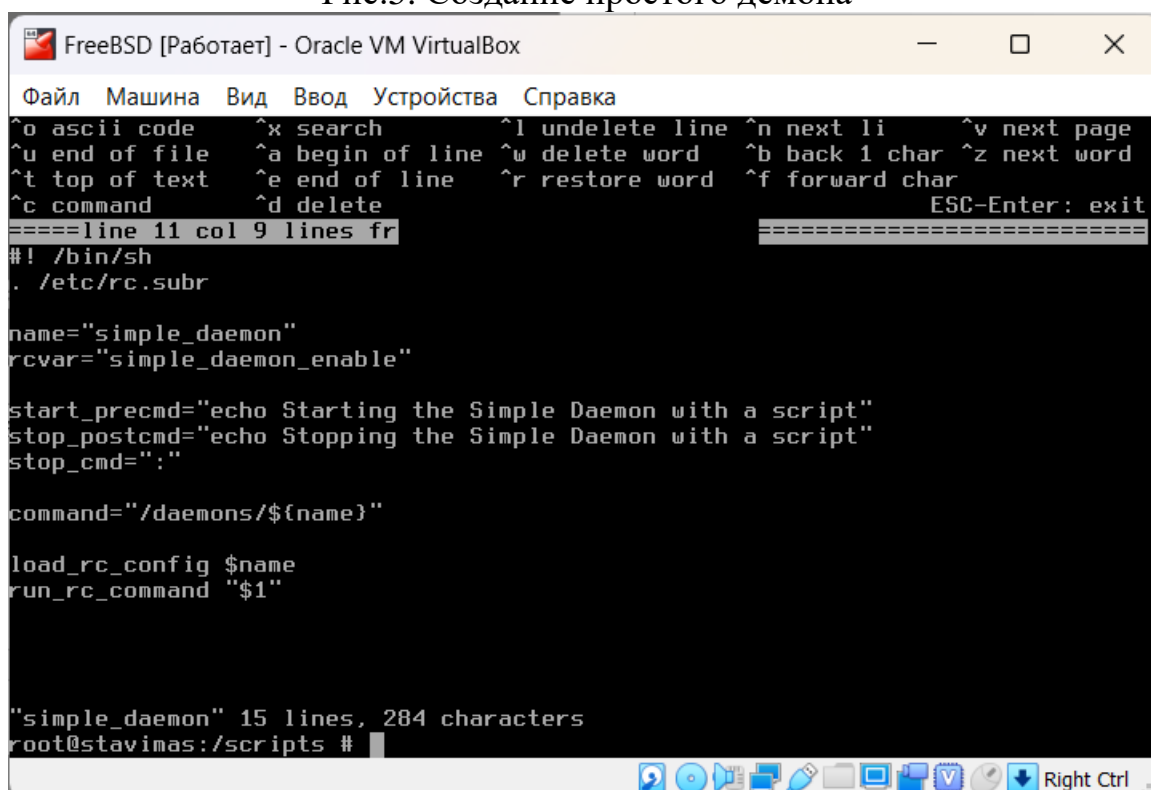


Рис.6. Создание rc.d скрипта для управления простым демоном

```
root@root:/scripts # sh simple_daemon start
Starting the Simple Daemon with a script
Starting simple_daemon.
Simple Daemon started successfully
```

Рис.7. Демонстрация работы демона, запускаемого с помощью rc.d скрипта

```

^o ascii code      ^x search          ^l undelete line  ^n next li        ^v next page
^u end of file     ^a begin of line   ^w delete word    ^b back 1 char    ^z next word
^t top of text     ^e end of line     ^r restore word   ^f forward char
^c command         ^d delete          ESC-Enter: exit
=====line 15 col 0 lines fr=====
# PROVIDE: simple_daemon
# REQUIRE: daemon
# BEFORE: login
# KEYWORD: nojall

name="simple_daemon"
rcvar="simple_daemon_enable"
pidfile="/daemons/${name}.pid"

start_precmd="echo Starting the Simple Daemon with a script"
stop_postcmd="echo Stopping the Simple Daemon with a script"
stop_cmd=":"

command="/daemons/${name}"

load_rc_config $name
run_rc_command "$1"

"simple_daemon" 21 lines, 392 characters
root@root:/scripts #

```

Рис.8. Доработка скрипта для его присоединения к инфраструктуре

```

root@root:/scripts # cp simple_daemon /etc/rc.d
root@root:/scripts #

```

Рис.9. Помещение скрипта в директорию /etc/rc.d для автозагрузки

```

exiting.
/etc/rc: WARNING: failed to start dhcpd
Mounting late filesystems:.
Configuring vt: keymap blanktime.
Starting squid.
Performing sanity check on sshd configuration.
Starting sshd.
Starting the Simple Daemon with a script
Starting simple_daemon.
Simple Daemon started successfully
Starting cron.
Starting background file system checks in 60 seconds.

```

Рис.10. Демонстрация запуска скрипта вместе со стартом системы

```

^[( (escape) menu  ^y search prompt  ^k delete line    ^p prev li        ^g prev page
^o ascii code      ^x search          ^l undelete line  ^n next li        ^v next page
^u end of file     ^a begin of line   ^w delete word    ^b back 1 char    ^z next word
^t top of text     ^e end of line     ^r restore word   ^f forward char
^c command         ^d delete char     ^j undelete char  ESC-Enter: exit
=====line 31 col 19 lines from top 31=====
numbled_stop() {
    if [ -e "$Pidfile" ] ; then
        kill -s TERM `cat ${pidfile}`
    else
        echo "${name} is not running"
    fi
}

numbled_status() {
    if [ -e "${pidfile}" ] ; then
        echo "${name} is running as pid `cat ${pidfile}` "
    else
        echo "${name} is not running"
    fi
}

load_rc_config $name
run_rc_command "$1"

```

Рис.11. Создание сложного демона

```

root@root:~ # /etc/rc.d/numbled start
root@root:~ # /etc/rc.d/numbled status
numbled is running as pid 1269
root@root:~ # /etc/rc.d/numbled stop
root@root:~ # /etc/rc.d/numbled status
numbled is not running
root@root:~ # █

```

Рис.12. Запуск сложного демона

```

^I (escape) menu  ^y search prompt ^k delete line   ^p prev li      ^g prev page
^o ascii code    ^x search        ^l undelete line ^n next li      ^v next page
^u end of file   ^a begin of line ^w delete word   ^b back 1 char ^z next word
^t top of text   ^e end of line   ^r restore word  ^f forward char
^c command       ^d delete char   ^j undelete char
=====line 40 col 20 lines from top 40 =====
squid_enable="YES"

defaultrouter="192.168.84.208"
dhcpd_enable="YES"
dhcpd_ifaces="YES"
named_enable="YES"

firewall_enable="YES"
firewall_type="open"
firewall_logging="YES"

squid_enable="YES"
simple_daemon_enable="YES"

numbled_enable="YES"█

```

Рис.13. Добавление сценариев к инфраструктуре (файл rc.conf)

Вывод: в ходе выполнения данной лабораторной работы были закреплены практические навыки по настройке основных сервисов системы FreeBSD.

Ответы на контрольные вопросы:

1. Объясните, что такое rc.d.

Это модульный скрипт запуска системы.

2. Объясните, что такое сценарий.

Скрипт — это понятие в программировании, обозначающее последовательность команд для выполнения конкретных операций. По сути, это небольшая программа, заточенная под определенное действие.

3. Объясните, что такое демон.

Демон — компьютерная программа в UNIX-подобных системах, запускаемая самой системой и работающая в фоновом режиме без прямого взаимодействия с пользователем.

4. Раскройте область применения демонов.

Демоны в Unix-системах выполняют ряд важных функций, включая:

- Сетевые службы: Демоны могут предоставлять сетевые службы, такие как веб-серверы, FTP-серверы, DNS-серверы и т.д. Эти службы работают в фоновом режиме и готовы к обработке запросов от клиентов в любое время.

- Резервное копирование: Демоны также используются для резервного копирования файлов и системных данных. Например, демон cron запускает задачи резервного копирования в заданные промежутки времени.
- Мониторинг: Демоны могут быть настроены для мониторинга системных ресурсов, таких как использование процессора, памяти и дискового пространства. Они могут предупреждать администратора системы, если ресурсы исчерпываются или если происходят ошибки.
- Системный журнал: Демон syslog отвечает за запись системных журналов, в которых хранятся сообщения об ошибках, событиях и других важных сведениях. С помощью этих журналов можно отслеживать работу системы и быстро реагировать на проблемы.
- Автоматическая настройка: Демоны также могут использоваться для автоматической настройки системы и ее компонентов. Например, демон udev отвечает за автоматическое определение и настройку новых устройств, подключенных к системе.

5. Перечислите команды для работы со сценариями.

chmod, echo, if и т.д.

6. Раскройте суть аргументов сценария.

Аргументы в сценарии - это значения, переданные скрипту при его запуске. Аргументы обычно представлены в виде строк и могут быть использованы внутри сценария для выполнения различных задач.

Аргументы передаются скрипту через командную строку. Каждый аргумент разделяется пробелом и может содержать любые символы, включая пробелы, кавычки и специальные символы.

7. Опишите назначение sh.

sh - это командный интерпретатор для Unix-подобных систем, который используется для выполнения командных скриптов. Он является стандартным оболочкой командной строки для большинства Unix-подобных операционных систем.

8. Раскройте смысл PID.

PID - это сокращение от "Process Identifier" (идентификатор процесса). Это уникальный числовой идентификатор, который операционная система присваивает каждому процессу, запущенному на компьютере.

PID используется для идентификации процесса в системе, например, чтобы управлять им или завершить его выполнение. Когда процесс запускается, операционная система присваивает ему уникальный PID. PID может быть использован для определения статуса процесса, в том числе для определения, работает ли процесс, завершился ли он, или возникли ли какие-либо проблемы во время его выполнения.

9. Объясните, зачем нужен Makefile.

Makefile - это текстовый файл, который содержит инструкции для автоматической компиляции, сборки и установки программного обеспечения. Он используется для автоматизации процесса сборки программного обеспечения и упрощения процесса разработки.

В Makefile содержатся правила для компиляции и сборки исходного кода, а также для генерации различных файлов, таких как документация, исполняемые файлы и т.д. При запуске команды make в командной строке, он автоматически обрабатывает Makefile и выполняет необходимые действия для сборки программы или проекта.

10. Перечислите преимущества гибких скриптов.

Гибкие скрипты имеют несколько преимуществ, включая:

1. Автоматизация повторяющихся задач: Гибкие скрипты помогают автоматизировать повторяющиеся задачи, что позволяет сэкономить время и уменьшить вероятность ошибок.
2. Удобство и простота использования: Гибкие скрипты легки в использовании и не требуют значительных знаний программирования. Они позволяют легко настраивать их поведение и обеспечивают возможность быстро и легко выполнять задачи.
3. Адаптируемость: Гибкие скрипты позволяют быстро и легко адаптировать поведение программы к изменяющимся условиям и требованиям.
4. Переносимость: Гибкие скрипты могут выполняться на различных платформах и операционных системах, что обеспечивает удобство использования в различных средах.
5. Возможность интеграции: Гибкие скрипты могут использоваться для интеграции различных программных систем и утилит, что позволяет упростить и ускорить процесс работы.