МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ автоматики и вычислительной техники

**ОПЕРАЦИОННЫЕ СИСТЕМЫ вычислительных систем в информационных технологиях.**

Методические указания для выполнения лабораторных работ для бакалавров II курса направления 09.03.03 Прикладная информатика

Очной формы обучения

профиль Прикладная информатика в экономике

НОВОСИБИРК 2019

В методических указаниях приведены подготовительные материалы, задания и методические указания для студентов по выполнению лабораторных работ.

Методические указания составлены в соответствии с Положением об организации и осуществлении образовательной деятельности по образовательным программам высшего образования – программам бакалавриата, программам специалитета, программам магистратуры в Новосибирском государственном техническом университете (НГТУ), принятый ученым советом (протокол №9 от 30.09.2015), учебным планом и рабочей программой дисциплины.

Методические указания ориентированы на студентов 2 курсов, очной и заочной форм обучения, обучающихся по направлению *09.03.03, Прикладная информатика, профиль прикладная информатика в экономике.*. Выполнение лабораторных работ осуществляется с использованием лицензионного программного обеспечения и материально-технического оснащения факультета АВТ.

Составитель(и), старший преподаватель *В.А. Плеханов*

Рецензент д.т.н., доцент *Ю.А. Мезенцев*

Методические указания подготовлены кафедрой АСУ

# ВВЕДЕНИЕ

Лабораторные работы по дисциплине «Операционные системы» нацелены на углубление теоретических знаний, формирование практических умений и опыта деятельности в области использования и управления Операционными Системами в информационных технологиях.

По учебному плану *6842, Направление подготовки: 09.03.03 Прикладная информатика, профиль Прикладная информатика в экономике,* на лабораторные работы по дисциплине «Операционные Системы» предусмотрено 16 часов. Данный объем часов распределен на 4 лабораторных работы.

Лабораторные работы выполняются студентами на аудиторных занятиях в соответствии с их расписанием.

Преподаватель предварительно извещает студентов о теме и задании на очередную лабораторную работу.

По результатам выполнения лабораторных работ студенты оформляют отчеты о выполнении каждой лабораторной работы в электронной форме. Отчет формируется либо на аудиторном занятии, либо самостоятельно , в условиях самостоятельной, домашней работы*.*

Соблюдение установленных в разделе 2 требований к содержанию, структуре и оформлению отчета является обязательным.

Для выполнения лабораторных работ используется сервер с операционной системой Linux CentOS 7, размещенный в облаке НГТУ. В дальнейшей работе студенты получают навыки работы в Comand Line Interface(Интерфейсе Командной Строки), изучают и используют команды и утилиты ОС Linux, язык интерпретатора **bash** для выполнения лабораторных работ.

Таблица 1

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № п/п | Тема лабораторной работы | Количество часов | Формируемые компетенции  (в соответствии с РП) | | | | | |
| ОПК-2 | | ОК-7 | | ПК-23 | |
| Знать основные понятия теории операционных систем, практические основы работы в интерпретаторах команд и инструментальных средствах | Уметь использовать средства разработки ПО в профессиональной деятельности | Знать особенности профессионального развития личности | Уметь выстраивать индивидуальные образовательные траектории, профессиональный рост и карьеру | Знать основные методы и инструментальные средства для решения прикладных задач на основе системного подхода | Уметь применять математические методы и инструментальные средства анализа, оптимизации для решения задач управления ОС в профессиональной деятельности |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Лабораторная работа  № 1 | Работа в интерфейсе командной строки ОС GNU/Linux. Интерпретатор bash.Редактор vi. | 4 | + | + |  |  | + | + |
| Лабораторная работа  № 2 | Обработка текстовых данных в ОС Linux с помощью команд(утилит) и shell-скриптов интерпретатора bash | 4 | + | + |  |  | + | + |
| Лабораторная работа  № 3 | Разработка программы-калькулятор на языке интерпретатора bash | 4 | + | + |  |  | + | + |
| Лабораторная работа  № 4 | Технология IPC(межпроцессного взаимодействия) и разработка ПО в различных инструментальных средах. | 4 | + | + |  |  | + | + |

|  |  |  |
| --- | --- | --- |
| ИТОГО по УП | 16 |  |

# Теоретические основы.

**Рассматриваемые вопросы:**

1. Удаленное подключение к облачному учебному серверу ОС Linux CentOS 7.
2. Работа с документацией по командам интерпретатора bash.
3. Информация , необходимая для выполнения лабораторной работы(включая описание команд интерпретатора bash).
4. Использование консольного текстового редактора **vim.**
5. Создание скриптов(программ) на языке интерпретатора **bash.**
6. Пример выполнения лабораторной работы.
7. Требования к оформлению лабораторной работы.

Методические рекомендации**:**

1. Удаленное подключение к облачному учебному серверу ОС Linux CentOS 7.

1.1 Для подключения к указанному учебному серверу необходимо получить индивидуальный login и password у преподавателя.

1.2 Установить на свою рабочую станцию ПО Secure Shell(ssh-клиент), имеющее наименование putty. Получить дистрибутив нужно у преподавателя. При установке следуйте указаниям инсталлятора.

1.3 Настроить ssh-клиент.

Для этого запустить putty.exe

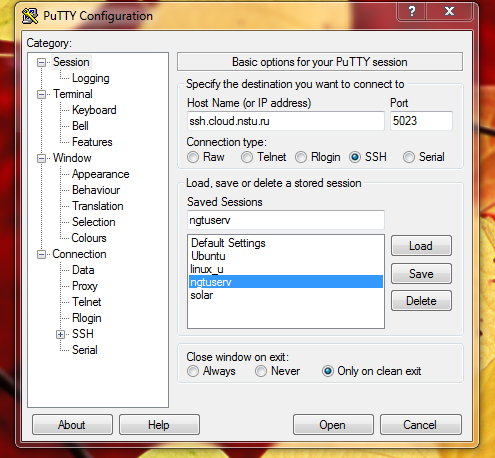


Рис.1 Окно настройки конфигурации программы Putty

- в окно “Host Name(or IP address)” ввести имя сервера, как указано на рисунке;

- в окно “Port” - 5023

- радиокнопка “Connection Type” должна быть выбранной в положение SSH;

- в окне “Saved Sessions” нужно ввести любое имя(например ngtuserv), для сохранения создаваемых данных соединения;

- нажать кнопку “Save”, после чего данные будут сохранены.

Далее нужно установить дополнительные параметры для данной сессии:

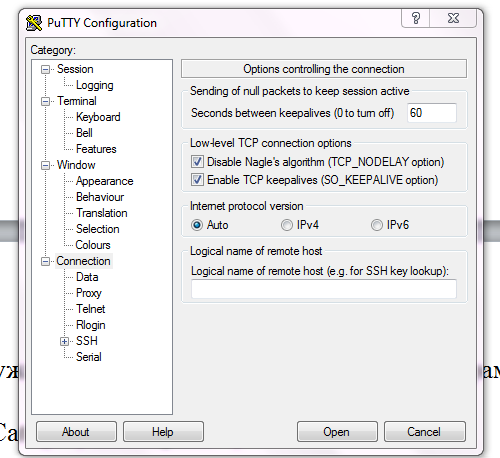


Рис.2 Окно настройки параметров программы Putty

-в окне “Category” выбрать пункт «Connection»;

- установить параметры , как указано на рис. 2;

- переключиться на пункт “Session” в окне “Category”(рис. 1);

- нажать на кнопку “Save”, правее окна “Saved Sessions”.

Клиент ssh(putty) настроен.

Для старта программы нужно выбрать сессию ngtuserv и нажать кнопку “Open”.

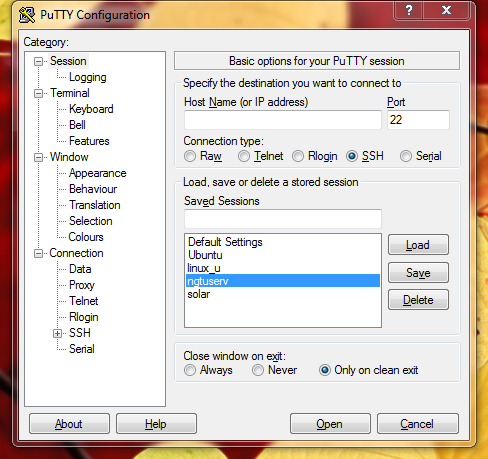


Рис. 3 Окно программы Putty для запуска соединения с сервером.

1. **Работа с документацией по командам интерпретатора bash и утилитам ОС.**

После аутентификации в ОС Linux CentOS, можно получить стандартную справку ОС UNIX/LINUX системы помощи **man.** Формат команды «man название команды». Примеры: **man login** – справочное руководство по команде login, **man man** – справочное руководство по команде man; **man bash** – справочное руководство по интерпретатору **bash**.

1. **Команды интерпретатора bash.**

После входа в ОС Linux, пользователь оказывается в директории /home/<Ваше\_Имя\_Пользователя>. Директория /home состоит из домашних каталогов пользователей. В своем домашнем каталоге пользователь имеет все права, т.е. он может выполнять операции без пароля администратора. Чтобы перейти из любого места файловой системы UNIX/Linux в свой домашний каталог, можно воспользоваться символом ~ вместе с командой **cd.** Пример :

**cd ~ (**аналогично работает вариант команды **cd** , без аргументов**)**

Значок **~** (тильда) - означает домашний каталог текущего пользователя.

**Список команд:**

**who** и **w** – информация о подключенных пользователях.

**whoami -** кто я(под каким пользователем я работаю в системе).

**env** – список значений переменных окружающей среды.

**ls** – список объектов текущего каталога.

**echo** - вывести значения аргументов команды echo на стандартный вывод (ключ **-n** означает, что символ конца строки не выводится).

**alias** – просмотр и назначение алиасов командам (в т.ч. командам с параметрами).

**unalias** – удаление алиасов.

**date** – вывод текущей даты

**clear** – очистка экрана.

**cd** (cddir) - изменение текущего каталога.

**chmod** – изменение атрибутов объектов файловой системы(файлы, каталоги).

**chown –** изменение собственника объекта файловой системы.

**chgrp -** изменение группы пользователей для объекта файловой системы.

**pwd** - где я ? полный путь к текущей директории.

**mkdir** - создание директории(каталога).

**rmdir -** удаление пустой директории.

**cp** - копирование объекта файловой системы.

**rm** - удаление объекта файловой системы.

**mv -** перемещение или переименование объекта файловой системы.

**cat** - вывести на экран содержимое файла (**cat** <имя файла> ) или ввод с консоли (**cat** >> <имя файла>). (закончить ввод - Ctrl+Z ).

**ps** - показать информацию о процессах в ОС.

**exit** – выход из текущего командного интерпретатора или скрипта.

**vi, vim** – вызов текстового редактора.

**touch –** создание нового файла нулевой длины.

**tr –** замена или удаление символов из входного потока.

**&&** - логический оператор И. Используется для объединения 2-х и более команд(операторов).

<Оператор 1> && <Оператор 2> - оператор2 будет выполнен только в случае успешного завершения оператора1 ,т.е. код завершения должен быть =“true”.

**|| -** логический оператор ИЛИ. Используется для объединения 2-х и более операторов.

<Оператор 1> || <Оператор 2> - оператор2 будет выполнен в случае ошибочного завершения оператора1 ,т.е. код завершения оператора1 должен быть =“false”.

**Управление вводом-выводом в консольных приложениях и утилитах ОС Linux.**

Каждый процесс ОС UNIX(Linux) имеет 3 открытых устройства(файла):

* stdin – стандартный ввод с клавиатуры(дескриптор файла = 0);
* stdout – стандартный вывод на экран(дескриптор файла = 1);
* stderr – стандартный вывод ошибок на экран(дескриптор файла = 2).

Перенаправления потоков данных осуществляются с помощью операций > , >> , < , |

* command  **>** file - стандартный вывод команды “command” перенаправляется в файл “file”. “file” создается заново. Например: tree > files
* command  **>>** file – стандартный вывод команды “command” перенаправляется в файл “file”. Информация дописывается в конец файла “file”.

Например: cat /var/log/events >> files

* command1  **|** command2 – стандартный вывод команды “command1” перенаправляется в стандартный ввод command2. Символ «|» - это pipe (конвеер) – однонаправленный канал межпроцессного взаимодействия.

Например: cat /var/log/events | grep error > errors

* command < file - стандартный ввод команды “command” перенаправляется из файла “file”.

Например: cat < files

Полезные мелочи:

Пользователи могут использовать автодополнение команд в некоторых командных интерпретаторах(включая bash). Для этого, начав ввод имени файла или директории в команде, нажмите TAB и , интерпретатор дополнит набираемое слово, при однозначном соответствии, либо предложит выбрать слово из списка.

1. **Использование консольного текстового редактора vim.**
2. **Создание скриптов(программ) на языке интерпретатора bash.**

**Shell-скрипт.**

Это текстовый файл(программа), состоящий из команд интерпретатора. Shell-скрипт выполняется командным интерпретатором – шеллом (sh,csh,ksh,bash). ОС UNIX/Linux выполняет только файлы программ с правом на выполнение. Право устанавливается с помощью команды

**chmod u+x имя\_скрипта**. В первой строке скрипта всегда указывается путь к интерпретатору, который должен выполнить данный скрипт- **#!/bin/bash**.

**Правила написания скриптов на языке интерпретатора bash.**

**Комментарии**. Оформляются символом # в начале строки, или в конце строки с командой интерпретатора(за исключением

комбинации #!).

**Пробелы и переводы строк.** Интерпретаторы команд чувствительны к пробелам и переводам строк. Отдельные команды должны начинаться с новой строки. При размещении нескольких команд в одной строке нужно разделять эти команды символом ‘;’. Параметры команды отделяются от команды пробелом, так же как и между собой.

**Особенности работы со строками**. Одиночные кавычки (' '),

ограничивающие строки с обеих сторон, не позволяют интерпретировать спец. символы . Будут представлены как обычные символы.

Внутри предложений, двойные кавычки (" ") позволяют интерпретировать специальные символы, $, ` (обратная кавычка) и \ (escape – обратный слэш), в соответствии со специальным смыслом .

Используйте двойные кавычки при обращении к переменным. Так же для экранирования специального символа можно применить обратного слэша ’\’. Он предотвращает интерпретацию следующего за ним символа.

**Переменные.** Именем переменной может быть

последовательность букв, цифр и подчеркиваний, начинающаяся с буквы или подчеркивания.

Пример:

val=2 # присваивание значения переменной

val2=$val # $val - обращение к значению переменной

Интерпретатор не имеет типов данных. Все переменные являются строковыми переменными. Однако интерпретатор может выполнять операции целочисленной арифметики

Например:

**let** val=2+2

или

val=”$((2+2))”

**Арифметические операторы:**

"+" сложение

"–" вычитание

"\*" умножение

"/" деление (целочисленное)

"\*\*" возведение в степень

"%" остаток от деления

**Оператор присваивания** ‘=’. В операциях присваивания нельзя ставить пробелы слева и справа от знака равенства.

**Специальные переменные.**

Список зарезервированных переменных интерпретатора bash:

$DIRSTACK – содержимое вершины стека каталогов.

$EUID – эффективный UID.

$UID – реальный идентификатор, который устанавливается

при логине.

$GROUPS – массив групп, к которым принадлежит текущий пользователь.

$HOME – домашний каталог пользователя.

$HOSTNAME – hostname компьютера.

$HOSTTYPE – архитектура машины.

$PWD – рабочий каталог.

$OSTYPE – тип ОС.

$PATH – путь поиска программ.

$PPID – идентификатор родительского процесса.

$SECONDS – время работы скрипта (в секундах).

$# – общее количество параметров, переданных скрипту.

$\* – все аргументы, переданные скрипту.

$@ – все аргументы, переданные скрипту.

$! – PID последнего запущенного в фоне процесса.

$$ – PID самого скрипта.

 Позиционные параметры. Аргументы, передаваемые скрипту из командной строки, хранятся в зарезервированных переменных $0, $1, $2, $3...,

где $0 – это название файла сценария, $1 – это первый аргумент, $2 –

второй, $3 – третий и так далее. Аргументы, следующие за $9, должны

заключаться в фигурные скобки, например: ${10}, ${11}, ${12}.

Передача параметров скрипту происходит в виде перечисления этих

параметров после имени скрипта через пробел в момент его запуска.

**Код завершения.** Команда **exit** используется для выхода из скрипта при завершении работы сценария. Параметр для команды- - это некоторое значение, которое может быть проанализировано вызывающим процессом. Представляет из себя код возврата, в виде **exit** nnn, где nnn – это код возврата (число в диапазоне 0– 255).

**Оператор вывода. еcho** переменные\_или\_строки.

**Оператор ввода. read** имя\_переменной. Одна команда read может прочитать (присвоить) значения сразу для нескольких переменных. Если переменных в read больше, чем их введено (через пробелы), оставшимся присваивается пустая строка. Если передаваемых значений больше, чем переменных в команде read, то лишние игнорируются.

**Условный оператор.**

**if** выражение1

**then** команда2

[**else**

команда3]

**fi**

Если выражение *выражение1* после выполнения возвращает значение “true”, то далее выполнится *команда2* . Иначе выполнится *команда3* . Для корректного оформления синтаксиса *выражение1* нужно обрамлять следующим образом - [[ выражение ]]. Обязательно ставить пробелы между выражением и скобками, например:

**if** [[ "$val" –gt "$val2" ]]

**then** echo "val > val2"

**fi**

**Операции сравнения**:

Операции сравнения целых чисел:

-eq # равно

-ne # не равно

-lt # меньше

-le # меньше или равно

-gt # больше

-ge # больше или равно

Операции сравнения строк:

-z # строка пуста

-n # строка не пуста

== # строки равны

!= # строки не равны

< # меньше (сравниваются коды символов)

<= # меньше или равно (сравниваются коды символов)

> # больше (сравниваются коды символов)

>= # больше или равно (сравниваются коды символов)

! # отрицание логического выражения

-a,(&&) # логическое «И»

-o,(||) # логическое «ИЛИ»

**Множественный выбор.** Для множественного выбора может

применяться оператор case.

**case** переменная **in**

значение1 **)**

команда 1

;;

значение2 **)**

команда 2

;;

**esac**

Сравниваются значение *переменная* со *значениеN*. При совпадении выполняется *командаN*. Каждая команда должна заканчиваться в новой строке – ;;

**Цикл for.** Существует два способа задания цикла for.

1. Стандартный – **for** переменная **in** список\_значений; **do**;

команды; **done**.

Например:

**for** i **in** 0 1 2 3

**do**

echo $i

**done**

2. С-подобный

**for** ((i=0; c <=3; i++))

**do**

echo $i

**done**

**Цикл while: while** условие; **do**; команда; **done**. Синтаксис

записи условия такой же, как и в условном операторе.

Например:

i=0

**while** [ i –lt 3 ]

**do**

echo “I равно $i”

**let** i+=1

**done**

**Управление циклами.** Управление ходом выполнения цикла осуществляется с помощью команд **break** и **continue**.

Команда break прерывает исполнение цикла, в то время как continue передает управление в начало цикла, минуя все последующие команды в теле цикла.

**Операторы работы со строками интерпретатора bash.**

${#string} – возвращает длину строки (string );

${string:position:length} – выделяет $length символов из строки

$string, начиная с позиции $position.

${string:position} извлекает подстроку из $string, начиная с позиции

$position.

${string#substring} – удаляет самую короткую из найденных подстрок

$substring в строке $string. Поиск ведется с начала строки. $substring

– регулярное выражение .

${string##substring} – удаляет самую длинную из найденных подстрок

$substring в строке $string. Поиск ведется с начала строки.

$substring – регулярное выражение.

${string/substring/replacement} – замещает первое вхождение

$substring строкой $replacement. $substring – регулярное выражение.

${string//substring/replacement} – замещает все вхождения

$substring строкой $replacement. $substring – регулярное

выражение.

**Работа со строками (внешние команды и утилиты).**

Для каждой команды доступно управление с помощью передаваемых команде параметров. Рекомендуем ознакомиться с документацией по этим командам с помощью команды man.

**sort** – сортирует поток текста в порядке убывания или возрастания, в

зависимости от заданных опций.

**uniq** – удаляет повторяющиеся строки из отсортированного файла.

**cut** – извлекает отдельные поля из текстовых файлов (поле –

последовательность символов в строке до разделителя).

**head** – выводит начальные строки из файла на stdout.

**tail** – выводит последние строки из файла на stdout.

**wc** – подсчитывает количество слов/строк/символов в файле или в потоке

**tr** – заменяет одни символы на другие.

**grep** pattern [file...] – утилита поиска участков текста в файле(ах),

соответствующих шаблону pattern, где pattern может быть как обычной

строкой, так и регулярным выражением.

**Sed** – неинтерактивный "потоковый редактор". Принимает текст либо с

устройства stdin, либо из текстового файла, выполняет некоторые операции над строками и затем выводит результат на устройство stdout или в файл.

Sed определяет, по заданному адресному пространству, над какими строками следует выполнить операции. Адресное пространство строк задается либо их порядковыми номерами, либо шаблоном. Например, команда 3d заставит sed удалить третью строку, а команда /windows/d означает, что все строки, содержащие "windows", должны быть удалены. Наиболее часто используются команды p – печать (на stdout), d – удаление и s – замена.

**awk** – утилита контекстного поиска и преобразования текста, инструмент для извлечения и/или обработки полей (колонок) в структурированных текстовых файлах. Awk разбивает каждую строку на отдельные поля. По умолчанию поля –

это последовательности символов, отделенные друг от друга пробелами, однако имеется возможность назначения других символов в качестве разделителя полей.

1. **Пример выполнения лабораторной работы.**

#!/bin/bash

# Лабораторная работа №1 (Задача 1)

#

b=3

c=2

let a=$b\*$c

g="$(($b+$c))"

f="$(($b-$c))"

r="$((2\*2))"

echo "значение переменной b= $b "|tr -d '\n\r'

echo "значение переменной c= $c "|tr -d '\n\r'

echo 'значение переменной а= b\*c= '$a

echo "значение переменной f= $f "

echo "значение переменной g= $g "

echo "значение переменной r= $r "

**Разработка программ в ОС Linux на языке «С».**

Компиляцию и компоновку программу необходимо выполнять с помощью GNU Compiler Collection.

GNU compiler collection (GCC) – это инструментальное средство разработки программ на языках Си, Си++, Фортран и пр. GCC состоит из:

- Препроцессоры программ на языках Си и Си++.

- Компиляторы для языков, которые входят в этот пакет. Компилятор – это программа, выдающая после обработки исходного текста программ , текст программы на языке ассемблера.

- Стандартные библиотеки языков Си++, С и других языков.

Команда запуска GCC для языка Си:

**gcc** <параметры> <файл с текстом программы>

Пример: **gcc pgm1.c pgm2.c -o pgm**

Или **gcc pgm1.c -o pgm**

Компилируются две программы на языке “С” – **pgm1 pgm2**, исполняемый файл будет носить имя - **pgm**. Файлы с текстами программ должны иметь расширение **.с**

**Системные вызовы.**

**Системный вызов open():**

Открытие файла и получение файлового дескриптора.

**Флаги для open()**: O\_RDONLY, O\_WRONLY или O\_RDWR.:

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

int open (const char \*name, int flags);

int open (const char \*name, int flags, mode\_t mode);

Пример:

int fd;

fd = open ("/home/kidd/madagascar", O\_RDONLY);

if (fd == –1)

fprintf (stderr, "Error open file on %d!\n", fd);

**Системный вызов close()** -Закрытие файла**.**

#include <unistd.h>

int close (int fd);

Пример:

int fd;

fd = open ("/home/st1/file1", O\_RDONLY);

if (close (fd) == –1)

fprintf (stderr, "Error close file on %d!\n", fd);

**Системный вызов creat():**

Системный вызов для создания файла:

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

int creat (const char \*name, mode\_t mode);

Пример:

int fd;

fd = creat (filename, 0644);

if (fd == –1)

fprintf (stderr, "Error create file on %s!\n", filename);

**Системный вызов remove() -** удаление файла**:**

#include <stdio.h>

int remove (const char \*path);

Пример:

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <stdio.h>

int fd;

fd = remove(“/home/st/file1”);

if (fd == -1)

{

fprintf (stderr, "Ошибка удаления файла %s\n", argv[1]);

}

}

**Системный вызов rename()** – перемещение файла:

#include <stdio.h>

int rename (const char \*oldpath, const char \*newpath);

Пример:

#include <stdio.h>

int ret;

ret = rename(“file1”,”file2”);

if (ret == -1)

{

fprintf (stderr, "Ошибка перемещения файла file1\n");

}

else {

fprintf(stderr,"файл file1 переименован в file2\n");

}

**Дополнительные условия:**

- При разработке программы студенты должны использовать консольный текстовый редактор vi(vim), либо nano.

- Перед разработкой программы студенты должны написать “Постановку задачи”, в которой предусмотреть ситуации, которые необходимо обрабатывать в программах, описать интерфейс программ с пользователем, составить блок-схему работы программ.

- В своей работе студенты могут пользоваться информацией из сети ИНТЕРНЕТ, могут общаться между собой.

1. **Список заданий для лабораторной работы №1.**

Цель работы:

Получить опыт интерактивной работы в файловой системе LINUX, используя интерфейс командной строки. Получить опыт работы в текстовом редакторе vi(vim). Получить начальный опыт программирования в среде интерпретатора bash.

1. Очистите домашнюю директорию от вложенных директорий.

2. Создайте каталоги в домашней директории documents и scripts.

3 . Далее создайте директории в соответствии с блок-схемой .

/home/student[1-12]/

documents/

scripts/

lab1/

labs/

lab1/

4. В директории /home/student[1-12]/scripts/lab1 напишите скрипты, решающие следующие задачи(имена файлов-скриптов давать в соответствии с буквой задания). Для создания скриптов используйте редактор vi(vim). Не забывайте вставлять комментарии в скрипты:

a) В параметрах при запуске скрипта передаются две любые строки, на ваш выбор. Вывести наименование скрипта,значение обоих параметров а так же сообщение о равенстве или неравенстве переданных строк. Повторить скрипт с выводом информации в файл. Файл назвать outa.

b) В параметрах при запуске скрипта передаются два целых числа. Вывести оба параметра.Вывести максимальное из них. Повторить скрипт с выводом информации в файл. Файл назвать outb.

c) Скопировать предыдущий скрипт в новый и модифицировать новый для обработки 4 чисел (параметров).Найти наибольшее число из 4-х.

Использовать только оператор if и оператор &&

Повторить скрипт с выводом информации в файл. Файл назвать outс.

5. Отчет оформить в MS Word и передать преподавателю.

1. **Задание для лабораторной работы №2.**

**Цель работы**: Разработать и изучить взаимодействие программ написанных на разных языках программирования в рамках ОС Linux, с использованием системных вызовов к ядру системы.

Разработка 2-ух взаимодействующих программ, написанных с использованием языков:

1. Интерпретатора bash;
2. «C»

Интерфейс с пользователем программы должен быть написан на языке интерпретатора bash и представлять из себя несколько пунктов интерактивного меню.

Например:

1. Описание программы.
2. Создание файла.
3. Удаление файла.

После выбора пункта меню пользователем, инициируется запрос на ввод необходимых для выполнения команды данных. Далее должен следовать вызов на исполнение программы написанной на языке «С» и скомпилированной/собранной в исполняемый модуль. Для каждого пункта меню - свой модуль.

**Варианты задания:**

1-вариант:

Операции:

- создание файла;

- удаления файла;

2-вариант:

Операции:

- создание файла;

-переименование файла;

3-вариант:

Операции:

- создание файла;

- копирование файла;

1. **Отчет о выполнении лабораторной работы.**

Студент обязан предоставить отчет о выполнении лабораторной работы не позднее времени ее окончания.

Содержание отчета должно быть полным, достоверным, логичным и аргументированным.

Структура отчета включает в себя:

1-Титульный лист.

2-Цель и задачи лабораторной работы.

3-Описание используемого оборудования.

4-Описание хода выполнения (порядок действий) лабораторной работы.

5-Результаты (расчеты, графики, таблицы и т.п.)

6-Ответы на контрольные вопросы.

7-Выводы (результаты обработки исходных данных и оценка полноты решения поставленной задачи).

1. **Требования к оформлению отчета о выполнении лабораторной работы:**

- титульный лист установленной формы (Приложение);

- текст должен содержать заголовки в соответствии со структурой отчета;

- формат А4;

- шрифт – Times New Roman;

- размер шрифта – 12-14 пт;

- цвет шрифта – черный;

- межстрочный интервал – 1-1,5;

- размеры полей: верхнее – 20 мм, нижнее – 20 мм, левое – 30 мм, правое – 10 мм;

- абзацный отступ – 1,25 мм;

- выравнивание текста – по ширине;

- страницы работы нумеруются арабскими цифрами в нижней части листа, начиная с титульного листа, номер на котором не ставится.

Отчет может быть представлен:

- в электронной форме;

- в распечатанном виде (по требованию преподавателя).

1. **Порядок, показатели , критерии и шкалы оценки сформированности компетенций.**

Студентом формируется отчет по каждой выполненной лабораторной работе.

Отчеты предоставляются преподавателю для проверки и оценивания.

Если преподаватель считает целесообразным, то он может пригласить студента на индивидуальное собеседование для оценки его знаний и практических навыков по вопросам и заданиям определенной лабораторной работы.

Защита. осуществляется во время лабораторной работы *из расчета 20 минут на защиту 1 студента*

Результаты выполнения лабораторных работ учитываются как элемент фиксации хода освоения дисциплины и отражаются:

1. в соответствии с Положением о балльно-рейтинговой системе оценки достижений студентов НГТУ (утверждено ректором 02.07.2009);
2. заносятся преподавателем в журнал текущей успеваемости по каждой из выполненных студентом лабораторных работ.

Максимальное количество баллов за выполнение всех лабораторных работ составляет 30 баллов *.*

Поскольку лабораторные работы формируют несколько компетенций, то преподавателем оценивается сформированность каждой из них по показателям, критериям и шкалам, приведенным в таблице 3.

В том случае если лабораторная работа не зачтена, студент обязан ознакомиться с замечаниями преподавателя, проанализировать допущенные ошибки, устранить выявленные недочеты и повторно сдать отчет на кафедру для проверки.

Результаты текущего контроля успеваемости учитываются преподавателем при проведении промежуточной аттестации. Отставание студента от графика текущего контроля успеваемости по изучаемой дисциплине приводит к образованию текущей задолженности.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Формируемые компетенции** | **Показатели** | **Критерии** | **Шкалы** | |
| **количество баллов БРС** | **оценка для текущего контроля** |
| **1** | **2** | **3** | **4** | **5** |
| ОПК-2 | |  | | --- | | Знать основные понятия теории операционных систем. | | Теоретический материал освоен глубоко и в полном объеме | 7 | отлично |
| Теоретический материал освоен на базовом уровне | 5 | хорошо |
| Теоретический материал освоен на уровне общего представления | 2 | удовлетворительно |
| Студент демонстрирует незнание значительной части теоретического материала | 0 | неудовлетворительно |
| Уметь использовать знания операционных систем в профессиональной деятельности | Теоретический материал освоен глубоко и в полном объеме | 8 | отлично |
| Теоретический материал освоен на базовом уровне | 6 | хорошо |
| Теоретический материал освоен на уровне общего представления | 3 | удовлетворительно |
| Студент демонстрирует незнание значительной части теоретического материала | 0 | неудовлетворительно |
| ПК-23 | Знать основные методы и инструментальные средства для решения прикладных задач на основе системного подхода к ОС | Теоретический материал освоен глубоко и в полном объеме | 7 | отлично |
| Теоретический материал освоен на базовом уровне | 5 | хорошо |
| Теоретический материал освоен на уровне общего представления | 2 | удовлетворительно |
| Студент демонстрирует незнание значительной части теоретического материала | 0 | неудовлетворительно |
| Уметь применять методы и инструментальные средства операционных систем для решения  любых задач в ОС. | Теоретический материал освоен глубоко и в полном объеме | 8 | отлично |
| Теоретический материал освоен на базовом уровне | 5 | хорошо |
| Теоретический материал освоен на уровне общего представления | 3 | удовлетворительно |
| Студент демонстрирует незнание значительной части теоретического материала | 0 | неудовлетворительно |
| МАКСИМАЛЬНОЕ КОЛИЧЕСТВО БАЛЛОВ | | | 30 |  |