МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Ярославский государственный университет им. П.Г. Демидова

Кафедра вычислительных и программных систем

УТВЕРЖДАЮ

Декан факультета ИВТ
_____ Д.Ю. Чалый
(подпись)

«_18__» __мая_____ 2021__ г.

Рабочая программа дисциплины
«Операционные системы»

Направление подготовки
09.03.03 Прикладная информатика

Направленность (профиль)
«Информационные технологии в цифровой экономике»

Форма обучения
очная

Программа одобрена
на заседании кафедры
от «23» апреля 2021 года, протокол № 8

Программа одобрена НМК
Факультета ИВТ
протокол № 7 от « 17 » мая 2021года

Ярославль
2021

1. Цели освоения дисциплины

Целями дисциплины «Операционные системы» являются изучение принципов устройства POSIX-совместимых операционных систем, приёмов и методики их администрирования. Поскольку операционные системы семейства UNIX в существенной степени разрабатывались в университетской среде и в чистом виде воплотили в себе многие основополагающие концепции построения компонентов операционных систем, то данный курс способствует фундаментализации образования.

2. Место дисциплины в структуре образовательной программы

Дисциплина «Операционные системы» относится к обязательной части ОП бакалавриата.

Она базируется на знаниях и навыках, полученных студентами при изучении общепрофессиональных дисциплин компьютерного цикла, в наибольшей степени дисциплины «Алгоритмизация и программирование».

Помимо расширения общепрофессиональной составляющей образования студентов дисциплина направлена на их подготовку к профессиональной деятельности в области системного администрирования и системной интеграции.

3. Планируемые результаты обучения по дисциплине, соотнесенные с планируемыми результатами освоения образовательной программы

Процесс изучения дисциплины направлен на формирование следующих элементов компетенций в соответствии с ФГОС ВО, ООП ВО и приобретения следующих знаний, умений, навыков и (или) опыта деятельности:

Формируемая компетенция (код и формулировка)	**Индикатор достижения компетенции (код и формулировка)**	**Перечень планируемых результатов обучения**
Общепрофессиональные компетенции		
ОПК-2. Способен понимать принципы работы современных информационных технологий и программных средств, в том числе отечественного производства, и использовать их при решении задач профессиональной деятельности;	ОПК-2.1 Обладает навыками выбора современных программных средств для решения прикладных задач в профессиональной деятельности.	Знать •принципы организации и операционных систем ; •устройство основных компонентов операционных систем. Уметь •выполнять основные операции с элементами файловой системы; •конфигурировать основные серверные службы; •выполнять типовые операции по обслуживанию операционной системы; Владеть •навыками разработки средств автоматизации (скриптов), предназначенных для решения типовых задач администрирования; • навыками планирования конфигурации и профилей настройки операционной системы в соответствии с заданными требованиями к её функционированию.

	ОПК-2.2 разрабатывает и реализует алгоритмы решения прикладных задач с использованием оригинальных алгоритмов и программных средств с использованием современных систем программирования	Знать •принципы организации и операционных систем семейства UNIX; •устройство основных компонентов операционных систем семейства UNIX; •основные команды операционных систем семейства UNIX. Уметь •выполнять основные операции с элементами файловой системы; •выполнять типовые операции по обслуживанию операционной системы UNIX; •диагностировать и устранять неполадки операционной системы UNIX; Владеть •навыками разработки средств автоматизации (скриптов), предназначенных для решения задач администрирования UNIX; навыками планирования конфигурации и профилей настройки операционной системы UNIX.
	ОПК-2.3 демонстрирует умение отобрать и/или адаптировать существующие информационные технологии и программные средства для конкретной прикладной задачи.	Знать: • структуру технической литературы и документации операционных систем в том числе в сети Интернет. Уметь: • использовать техническую документацию операционной системы UNIX. Владеть: навыками поиска информации об операционной системе UNIX.

4. Объем, структура и содержание дисциплины

Общая трудоемкость дисциплины составляет 4 зачетных единиц, 144 акад. часов.

№ п/п	Темы (разделы) дисциплины, их содержание	Семестр	Виды учебных занятий, включая самостоятельную работу студентов, и их трудоемкость (в академических часах)						Формы текущего контроля успеваемости Форма промежуточной аттестации *(по семестрам)* ***Формы ЭО и ДОТ (при наличии)***
			Контактная работа					самостоятельная работа	
			лекции	практические	лабораторные	консультации	аттестационные испытания		
1	История, базовые понятия и механизмы ОС UNIX	3	2		4			5	
2	Базовые команды UNIX	3	2		4			5	лабораторная работа
	том числе ЭО и ДОТ							*5*	
3	Устройство файловой системы и команды для работы с файлами	3	2		4			5	
4	Процессы в UNIX. Управление процессами	3	2		4			5	
5	Права доступа процессов к файлам и каталогам	3	2		4			5	
6	Командный интерпретатор bash. Разработка shell-скриптов	3	2		4			5	лабораторная работа
	том числе ЭО и ДОТ							*4*	
7	Текстовые редакторы vim и emacs	3	1		4			5	
8	Обработка текстовых данных в UNIX	3	1		1			5	
9	Процесс начальной загрузки системы (на примере ОС GNU/Linux)	3	3		1			4	Тест
		3							Экзамен
	ИТОГО		**17**		**34**	**2**	**0,5**	**53**	
	в том числе с ЭО и ДОТ							***9***	

Содержание разделов дисциплины:

1. История, базовые понятия и механизмы ОС UNIX.
2. Базовые команды UNIX
3. Устройство файловой системы и команды для работы с файлами.Разметка жёсткого диска, таблица разделов, расширенные и дополнительные разделы. Понятие виртуальной файловой системы и точек монтирования. Виды файловых систем (общего назначения: ext4, brtfs, zfs, reiserfs). Создание файловых систем и инструменты их создания. Дефрагментация файловых систем.
4. Процессы в UNIX. Управление процессами
5. Права доступа процессов к файлам и каталогам

6. Командный интерпретатор bash. Разработка shell-скриптов
7. Текстовые редакторы vim и emacs
8. Обработка текстовых данных в UNIX
9. Процесс начальной загрузки системы (на примере ОС GNU/Linux)

5. Образовательные технологии, в том числе технологии электронного обучения и дистанционные образовательные технологии, используемые при осуществлении образовательного процесса по дисциплине

В процессе обучения используются следующие образовательные технологии:

Вводная лекция – дает первое целостное представление о дисциплине и ориентирует студента в системе изучения данной дисциплины. Студенты знакомятся с назначением и задачами курса, его ролью и местом в системе учебных дисциплин и в системе подготовки в целом. Дается краткий обзор курса, история развития науки и практики, достижения в этой сфере, имена известных ученых, излагаются перспективные направления исследований. На этой лекции высказываются методические и организационные особенности работы в рамках данной дисциплины, а также дается анализ рекомендуемой учебно-методической литературы.

Академическая лекция с элементами лекции-беседы – последовательное изложение материала, осуществляемое преимущественно в виде монолога преподавателя. Элементы лекции-беседы обеспечивают контакт преподавателя с аудиторией, что позволяет привлекать внимание студентов к наиболее важным темам дисциплины, активно вовлекать их в учебный процесс, контролировать темп изложения учебного материала в зависимости от уровня его восприятия.

Практическое занятие – занятие, посвященное освоению конкретных умений и навыков по закреплению полученных на лекции знаний.

Консультации – вид учебных занятий, являющийся одной из форм контроля самостоятельной работы студентов. На консультациях по просьбе студентов рассматриваются наиболее сложные моменты при освоении материала дисциплины, преподаватель отвечает на вопросы студентов, которые возникают у них в процессе самостоятельной работы.

В процессе обучения используются следующие технологии электронного обучения и дистанционные образовательные технологии:

Электронный учебный курс «Операционные системы» в LMS Электронный университет Moodle ЯрГУ, в котором:

- представлены задания для самостоятельной работы обучающихся по темам дисциплины;
- осуществляется проведение отдельных мероприятий текущего контроля успеваемости студентов;
- представлены тексты лекций по отдельным темам дисциплины;
- представлены правила прохождения промежуточной аттестации по дисциплине;
- представлен список учебной литературы, рекомендуемой для освоения дисциплины;
- представлена информация о форме и времени проведения консультаций по дисциплине в режиме онлайн;
- посредством форума осуществляется синхронное и (или) асинхронное взаимодействие между обучающимися и преподавателем в рамках изучения дисциплины.

6. Перечень лицензионного и (или) свободно распространяемого программного обеспечения, используемого при осуществлении образовательного процесса по дисциплине

В процессе осуществления образовательного процесса по дисциплине используются:

для формирования материалов для текущего контроля успеваемости и проведения промежуточной аттестации, для формирования методических материалов по дисциплине:

- программы Microsoft Office;
- издательская система LaTex;
- Adobe Acrobat Reader.
- OS Linux (свободная)

7. Перечень современных профессиональных баз данных и информационных справочных систем, используемых при осуществлении образовательного процесса по дисциплине (при необходимости)

В процессе осуществления образовательного процесса по дисциплине используются:

Автоматизированная библиотечно-информационная система «БУКИ-NEXT» http://www.lib.uniyar.ac.ru/opac/bk_cat_find.php

8. Перечень основной и дополнительной учебной литературы, ресурсов информационно-телекоммуникационной сети «Интернет» (при необходимости), рекомендуемых для освоения дисциплины

а) основная литература

1. Робаческий, А. М., Операционная система UNIX / А. М. Робачевский, С. А. Немнюгин, О. Л. Стесник. - 2-е изд., перераб. и доп., СПб., БХВ-Петербург, 2014, 635с

б) дополнительная литература

1. Костромин, В. А., Самоучитель Linux для пользователя, СПб., БХВ-Петербург, 2004, 672с

2. Курячий Г. В. Операционная система Linux: курс лекций : учеб. пособие для вузов. / Г. В. 3. Курячий, К. А. Маслинский - М.: Интернет-Ун-т Информационных Технологий, 2005. - 387 с.

9. Материально-техническая база, необходимая для осуществления образовательного процесса по дисциплине

Материально-техническая база, необходимая для осуществления образовательного процесса по дисциплине включает в свой состав специальные помещения:

- учебные аудитории для проведения занятий лекционного типа;
- учебные аудитории для проведения практических занятий (семинаров);
- учебные аудитории для проведения групповых и индивидуальных консультаций;
- учебные аудитории для проведения текущего контроля и промежуточной аттестации;
- помещения для самостоятельной работы;
- помещения для хранения и профилактического обслуживания технических средств обучения.

Специальные помещения укомплектованы средствами обучения, служащими для представления учебной информации большой аудитории.

Помещения для самостоятельной работы обучающихся оснащены компьютерной техникой с возможностью подключения к сети «Интернет» и обеспечением доступа к электронной информационно-образовательной среде ЯрГУ.

Автор:

Доцент кафедры вычислительных и программных систем		А.М. Васильев
должность, ученая степень	*подпись*	*И.О. Фамилия*

Приложение № 1 к рабочей программе дисциплины
«Операционные системы»

Фонд оценочных средств
для проведения текущего контроля успеваемости
и промежуточной аттестации студентов
по дисциплине

1. Типовые контрольные задания и иные материалы, используемые в процессе текущего контроля успеваемости

1.1. Контрольные задания и иные материалы, используемые в процессе текущей аттестации

Примеры заданий для лабораторных работ

Лабораторная работа по теме «Базовые команды UNIX»

Целью данного задания является ознакомление с системными вызовами UNIX. В качестве практики предлагается реализовать простую командную оболочку. Результат своей работы (исходный код командной оболочки) будет необходимо загрузить в качестве ответа.

Скачайте скелет командной оболочки, прикреплённый к этому практическому заданию, и просмотрите его код. Исходный код можно разделить на две части: обработку строки команды и её выполнение. Обработкой занимается часть приложения, называемая парсером.

Парсер поддерживает разбор лишь для базовых команд. Пример набора команд приведён ниже. Скопируйте эти команды в файл с названием test.sh.:

```
ls > y
cat < y | sort | uniq | wc > y1
cat y1
rm y1
ls |  sort | uniq | wc
rm y
```

Вы можете скомпилировать скелет командной оболочки с помощью компилятора gcc::

```
$ gcc sh.c
```

В результате будет создан исполняемый файл с названием a.out, который следует запустить следующим образом::

```
$ ./a.out < test.sh
```

Во время выполнения вы увидите множество сообщений. Эти сообщения говорят о невозможности выполнить действия из скрипта.

Вашей задачей является модификация скелета командной оболочки таким образом, чтобы приложение могло успешно выполнять тестовый скрипт.

Выполнение простых команд

Начните реализацию данного приложения с поддержки выполнения простых команд, таких как::

```
$ ls
```

То есть ваша командная оболочка должна научиться запускать простые приложения.

Парсер уже может разбирать данную ситуацию. В результате разбора создаётся структура типа execcmd, которая передаётся функции runcmd.

Для решения этой задачи достаточно написать код обработчика ситуации ' ' внутри оператора case. Вам потребуется использовать системный вызов exec, поэтому прочитайте официальную документацию, доступную в системе руководств::
$ man 3 exec
Для проверки реализации в тестовом файле достаточно написать одну команду ls.

Перенаправление ввода-вывода

Далее реализуйте перенаправление вывода в файл и перенаправление ввода из файла. Данная операция реализуется в командной оболочке с помощью операторов > и <. После реализации этого функционала ваше приложение сможет выполнять следующий код::
echo "6.828 is cool" > x.txt
cat < x.txt
Парсер умеет разбирать команды перенаправления вывода в файл (>) и ввода из файла (<). Если команда содержала перенаправление, то создаётся структура типа redircmd. Как и в предыдущем случае достаточно реализовать код в соответствующей ветке оператора case.
Для реализации этого функционала вам потребуются системные вызовы open и close. Возможно, придётся посмотреть данную информацию в сети Интернет.
Для проверки используйте код примера, приведённого выше.

Реализация конвееров

Далее реализуйте перенаправление ввода-вывода с использованием конвееров (pipe), чтобы можно было выполнить следующий код::
$ ls | sort | uniq | wc
Парсе распознаёт команды по созданию конвеера. В результате разбора создаётся структура pipecmd. Для поддержки достаточно реализовать код в последней ветке оператора case: '|'.
Для реализации этой функции вам могут потребоваться системные вызовы pipe, fork, close, dup.
Для проверки используйте код примера, приведённого выше.

Последняя проверка

Убедитесь, что ваша командная оболочка способна выполнять код примера, приведённого в начале задания.
После этого можете загружать решение задачи на сайт.

Лабораторная работа по теме «Командный интерпретатор bash. Разработка shell-скриптов»

Ваша задача состоит в том, чтобы выводить строку с описанием системного вызова каждый раз, когда ядро обрабатывает системный вызов. Достаточно вывести лишь имя системного вызова и возвращаемое значение. Не надо выводить значения аргументов системного вызова.
В рамках данной задачи вам предлагается ознакомится с обработкой системных вызовов внутри ядра операционной системы.
После выполнения этой задачи, при загрузке операционной системы вы увидите строки, наподобие следующих:
fork -> 2
exec -> 0
open -> 3
close -> 0
$write -> 1
write -> 1
Данная последовательность системных вызовов описывает создание нового процесса системой инициализации init, и последующего запуска приложения sh. Данный процесс открывает два файловых дескриптора, а после sh записывает символы $ и пробел.

Для решения этой задачи вам потребуется изменить код функции syscall(), находящихся в файле syscall.c.

Дополнительное задание: разберитесь с системой получения аргументов системных вызовов и выведите их вместе с именем системного вызова.

Реализация нового системного вызова

В рамках данного задания предлагается реализовать системный вызов halt, позволяющий завершить работу операционной системы. Цель данного задания - изучить различные особенности организации системных вызовов.

Реализация системного вызова будет завершать работу виртуальной машины QEMU. Ниже приведён кусок исходного кода, который позволяет это сделать::

```
char *p = "Shutdown";
for( ; *p; p++)
   outb(0x8900, *p);
```

Данный код должен выполняться на уровне ядра операционной системы, а не в рамках прикладного программного обеспечения.

Помимо реализации сисемного вызова, вам потребуется реализовать программу прикладного уровня, которая будет делать системный вызов. Добавьте следующий код в файл halt.c ::

```
#include "types.h"
#include "stat.h"
#include "user.h"

int main(int argc, char *argv[])
{
  halt();
  return 0;
}
```

Для того, чтобы это приложение можно было вызывать из операционной системы Xv6, добавьте _halt к списку UPROGS, определённого в файле Makefile.

Предлагаемый подход

В качестве основы для создания нового системного вызова вам предлагается изучить и скопировать структуру существующего системного вызова. Наиболее подходящим является системный вызов, который не принимает никаких аргументов, например uptime.

Сначала вам необходимо выяснить: в каких файлах находится реализация поддержки системных вызовов. Для этого используйте приложение grep ::

```
$ grep -n uptime *.[chS]
```

После того, как закончите реализацию, вызов halt внутри операционной системы xv6 приведёт к завершению эмулятора.

Дополнительное задание: Реализуйте системный вызов dup2().

Критерии оценивания лабораторных работ

Оценка	Критерии
Отлично Уровень формирования компетенций: высокий	ОПК-2: Задание выполнено полностью, включая дополнительные пункты, если они сформулированы. Студент знает подходящие для решения средства и инструменты. Понимает использованные команды, свободно дает пояснения к выполненным действиям.
Хорошо Уровень	ОПК-2: Задание выполнено полностью, но без дополнительных пунктов, если они сформулированы или с небольшими

формирования компетенций: продвинутый	неточностями в одном – двух пунктах задания. Студент знает подходящие для решения средства и инструменты. Понимает использованные команды, дает пояснения к большинству выполненных действий.
Удовлетворительно Уровень формирования компетенций: пороговый	ОПК-2: Задание выполнено не полностью, или с ошибками, но не более чем в половине задания. Студент знает основные подходящие для решения средства и инструменты, но не для всех сформулированных пунктов задания. Понимает использованные команды, но с трудом дает пояснения к большинству выполненных действий.
Неудовлетворительно	ОПК-2: Задание не выполнено, или выполнено с грубыми ошибками, в большей части задания. Студент не знает подходящие для решения средства и инструменты. Не понимает использованные команды, не дает пояснения к большинству выполненных действий.

Примеры вопросов для тестового задания

Вопрос №1. Находясь в каталоге /var/log/messages Василий выполнил команду `ls -l el` и увидел следующий вывод:

lrwxrwxrwx 1 ad ad 20 дек 21 17:45 el -> ../lib/vagrant

Затем выполнил команду `ls -ld ../lib/vagrant` и увидел следующий вывод:

drwxr-xr-x 2 ad ad 4096 дек 3 15:38 ../vagrant

1.1. Чем является файл vagrant?

1.2. Как вы это определили?

1.3. Укажите полный путь к данному файлу.

1.4. Укажите полный путь к дополнительному файлу.

Ответ на вопрос №1.

1.1. Файл `vagrant` является директорией.

1.2. В выводе второй команды первая буква `d`, что обозначает, что файл является директорией.

1.3. Полный путь к каталогу `vagrant`: `/var/log/lib/vagrant`.

1.4. Полный путь к символической ссылке `el`: `/var/log/messages/el`.

Вопрос №2. Вы находитесь в каталоге /tmp. В нём существует каталог a, в котором есть пустой файл b.

2.1. Создайте каталог /tmp/c, в котором будет пустой каталог d.

2.2. Сделайте задание пункта 2.1 другим способом

2.3. Создайте символическую ссылку /tmp/d, ссылающуюся на пустой файл b.

Ответ на вопрос №2.

2.1.

```
$ mkdir c
$ mkdir c/d
```

2.2.

```
$ mkdir -p c/d
```

2.3.

```
ln -s /tmp/d b
```

Вопрос №3. Василий выполнил команду `ls -ld /usr/local/lib` и увидел следующий вывод:

drwxrwsr-x 5 root staff 57 сен 19 14:29 /usr/local/lib

3.1. Что означает буква s в разрешениях для группы?
3.2. Напишите все факты о данном файле, которые можете извлечь из данного вывода.

Ответ на вопрос №3.

3.1. Буква `s` в разрешениях файла говорит о том, что при создании файлов или каталогов внутри данного каталога они будут принадлежать группе `staff` вне зависимости от того какими правами обладает пользователь, создающий данные каталоги.

3.2.

- Данный файл является директорией, т.к. первая буква в разрешениях - `d`.
- Файл принадлежит пользователю `root` и группе `staff`.
- Владелец может выполнять все функции с данным каталогом: добавлять, удалять, переименовывать файлы и запускать в рамках каталога любые приложения.
- Такими же правами обладает и группа `staff`.
- Все остальные пользователи могут лишь просматривать содержимое данного каталога и запускать в нём приложения.

Список вопросов к экзамену:

1. Файловая подсистема
2. Разметка жёсткого диска, таблица разделов, расширенные и дополнительные разделы.
3. Понятие виртуальной файловой системы и точек монтирования.
4. Проверка доступного и занятого дискового пространства, приложения du, di, df, ls, brasero.
5. Использование программы fdisk.
6. Виды файловых систем (общего назначения: ext4, brtfs, zfs, reiserfs).
7. Создание файловых систем и инструменты их создания.
8. Проверка файловых систем с помощью fsck и badblocks.
9. Дефрагментация файловых систем. Инструмент e4defrag, shake, defrag.
10. Процессы в UNIX. Управление процессами
11. Права доступа процессов к файлам и каталогам
12. Командный интерпретатор bash. Разработка shell-скриптов
13. Текстовые редакторы vim и emacs
14. Обработка текстовых данных в UNIX
15. Процесс начальной загрузки системы (на примере ОС GNU/Linux)

Приложение № 2 к рабочей программе дисциплины
«Операционные системы »

Методические указания для студентов по освоению дисциплины

Занятия по данной дисциплине проводятся в различных формах. Все лекционные занятия проводятся в компьютерных классах с использованием мультимедиа-технологий, что позволяет выполнять немедленную демонстрацию концепций устройства UNIX-систем и принципов их администрирования, а также возможностей конкретных команд на практике, а также обеспечивает возможность изучения их студентом в интерактивном режиме. Практическое применение полученных знаний отрабатывается при выполнении лабораторных работ во время практических лабораторных занятий. Разбор типовых ошибок также осуществляется в ходе лабораторных занятий с привлечением метода мозгового штурма, активирующего креативные способности студентов.

Основной формой практической работы студентов по усвоению данного курса является выполнение ими самостоятельных лабораторных работ. По итогам каждой из лабораторных работ проводится промежуточная аттестация студента. Окончательная аттестация осуществляется в форме зачёта (в 5-м семестре) и экзамена (в 6-м семестре). Допуск к зачётам и экзаменам осуществляется в форме компьютерного тестирования, а также принимает во внимание средний балл по результатам промежуточных аттестаций.