Белорусский государственный университет

УТВЕРЖДАЮ
Проректор по учебной работе и
образовательным инновациям
_____ О.Н.Здрок
«4» января 2020 г.

Регистрационный № УД-7801/уч.

ВНУТРЕННЕЕ УСТРОЙСТВО ОС СЕМЕЙСТВА UNIX

Учебная программа учреждения высшего образования по учебной дисциплине для специальности:

1-31 80 09 Прикладная математика и информатика

профилизация
Алгоритмы и системы обработки больших данных

2020 г.

Учебная программа составлена на основе ОСВО 1-31 80 09-2019 и учебного плана G31-072/уч. от 11.04.2019 г.

СОСТАВИТЕЛЬ:

С.А. Соболь – старший преподаватель кафедры дискретной математики и алгоритмики факультета прикладной математики и информатики Белорусского государственного университета.

РЕЦЕНЗЕНТЫ:

А.А. Толстиков – старший преподаватель кафедры вычислительной математики факультета прикладной математики и информатики Белорусского государственного университета;
А.Ю. Рыжиков – программист ООО «Фитбит Бел».

РЕКОМЕНДОВАНА К УТВЕРЖДЕНИЮ:

Кафедрой дискретной математики и алгоритмики
(протокол № 9 от 26 декабря 2019 года);

Научно-методическим Советом БГУ

(протокол № 3 от 3 января 2020 года).

Заведующий кафедрой
дискретной математики и алгоритмики _____ В.М. Котов

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Цели и задачи учебной дисциплины

Учебная дисциплина «Внутреннее устройство ОС семейства UNIX» знакомит студентов магистратуры с принципами, заложенными в функционирование UNIX-подобных операционных систем, их архитектурой и основами системного программирования.

Цель учебной дисциплины – создание базы для использования студентами вычислительных систем под управлением UNIX-подобных ОС (в частности, как инфраструктуры в задачах обработки больших объемов данных). Дисциплина формирует у студентов понимание того, как функционирует операционная система на низком уровне. Такие знания необходимы для того, чтобы профессионально и эффективно использовать технологии и компьютерные системы, основанные на принципах UNIX, при решении задач, связанных с обработкой и анализом данных.

При изложении материала учебной дисциплины важно показать спектр применения UNIX-подобных операционных систем при решении прикладных задач обработки и анализа больших объемов информации, возникающих в различных областях науки, техники, экономики и др., в сравнении с альтернативами по таким критериям, как стоимость, удобство сопровождения, техническая поддержка и др.

Задачи учебной дисциплины:

1. Изучение внутреннего устройства доступных операционных систем семейства UNIX.
2. Получение навыков разработки системного программного обеспечения на языке программирования C.
3. Формирование общего представления о механизмах, лежащих в основе функционирования современных компьютеров.

Место учебной дисциплины в системе подготовки специалиста с высшим образованием (магистра).

Учебная дисциплина относится к модулю «Большие данные» компонента учреждения высшего образования.

Программа составлена с учетом **межпредметных связей** с учебными дисциплинами. Основой для изучения учебной дисциплины являются следующие учебные дисциплины первой ступени высшего образования: «Программирование», «Операционные системы», «Архитектура компьютеров», «Компьютерные сети». Кроме того, программа данной дисциплины логически продолжает, расширяет и дополняет программу дисциплины «Эксплуатация и администрирование UNIX-систем», изучаемой на второй ступени высшего образования в рамках модуля «Большие данные». Знания, полученные в учебной дисциплине, используются далее при изучении дисциплины «Технологии проектирования и разработки высоконагруженных веб-систем» модуля «Избранные главы компьютерных наук».

Требования к компетенциям

Освоение учебной дисциплины «Эксплуатация и администрирование UNIX-систем» должно обеспечить формирование следующих специализированных и универсальных компетенций.

специализированные компетенции:

СК-2 Обладать пониманием архитектуры операционных систем, организацией памяти процессов и способах их взаимодействия.

универсальные компетенции:

УК-5. Обладать способностью в минимальные сроки изучать и профессионально эксплуатировать программные системы, модули и библиотеки.

В результате изучения дисциплины студент магистратуры должен:

знать:

- базовую архитектуру операционных систем семейства UNIX;
- название, назначение и особенности ряда системных вызовов;
- организацию памяти процессов, способы взаимодействия процессов;
- иерархию протоколов в сетях TCP/IP;
- принципы работы файловой системы;

уметь:

- выполнять системные вызовы из кода на C и ассемблере x86-64;
- порождать новые процессы и ожидать их завершения;
- организовывать межпроцессное взаимодействие;
- запускать потоки и работать с примитивами синхронизации;
- выделять виртуальные страницы и отображать файлы в память;
- создавать разделяемые библиотеки и использовать их;
- разрабатывать сетевые приложения на базе сокетов;
- конфигурировать и собирать ядро ОС Linux;
- строить виртуальные окружения для изоляции приложений;

владеть:

- понятиями *процесс*, *поток*, *системный вызов*, *сигнал*, *неименованный канал*, *мьютекс*, *условная переменная*, *фьютекс*, *виртуальная память*, *таблица страниц*, *сокет* и др.;
- компьютерными системами, основанными на принципах UNIX.

Структура учебной дисциплины

Дисциплина изучается во втором семестре. Всего на изучение учебной дисциплины «Внутреннее устройство ОС семейства UNIX» отведено:

– для очной формы получения высшего образования – 126 часов, в том числе 40 аудиторных часов, из них: лекции – 20 часов, лабораторные занятия – 20 часов.

Трудоемкость учебной дисциплины составляет 3 зачетные единиц.

Форма текущей аттестации по учебной дисциплине – зачет.

СОДЕРЖАНИЕ УЧЕБНОГО МАТЕРИАЛА

Тема 1. Введение.

Основы языка программирования C. Базовые понятия и синтаксис. Стандартная библиотека. Компиляция и компоновка кода. Средства отладки и анализа программ.

Тема 2. Системные вызовы.

Реализация стандартной библиотеки языка C. Механизм выполнения системных вызовов на компьютерах архитектуры x86-64.

Тема 3. Ввод-вывод.

Файловые дескрипторы. Функции ввода/вывода из POSIX C API: open, close, read, write, lseek. Дублирование дескрипторов. Буферизация.

Тема 4. Сигналы.

Ситуации, приводящие к получению сигнала. Средства командной строки для посылки сигналов процессам. Реализация обработчика сигнала на языке C. Прерываемые системные вызовы.

Тема 5. Порождение процессов.

Системный вызов fork. Процессы-зомби и процессы-сироты. Техника fork — exec. Запуск произвольной программы в UNIX.

Тема 6. Неименованные каналы.

Реализация конвейера в командной оболочке. Проблема обнаружения ошибки вызова exec при создании нового процесса и метод её решения.

Тема 7. Потоки.

Стандарт POSIX Threads. Создание и завершение потока. Мьютексы и условные переменные.

Тема 8. Внутренняя реализация потоков.

Ручной запуск потока. Атомарные операции. Устройство примитивов синхронизации. Механизм futex.

Тема 9. Управление памятью.

Понятие виртуальной памяти и реализация на компьютерах архитектуры x86-64. Механизм копирования при записи (copy-on-write). Ситуации отказа страниц (page faults). Классификация виртуальной памяти: память private и shared, anonymous и file-backed. Ввод-вывод посредством отображения файлов в память. Разделяемая память: способы создания, использование для межпроцессного взаимодействия. Кеширование дисковых операций, сброс кешей. Ситуация нехватки памяти в системе. Swap-раздел.

Тема 10. Структура программы.

Формат ELF. Секции, сегменты, символы. Реализация разделяемых библиотек в Linux: загрузчик, механизм LD_PRELOAD. Проблема релокации.

Тема 11. Сети TCP/IP.

Протоколы IPv4 и IPv6. Изучение протокола TCP. Анализ трафика при помощи tcpdump. Протокол HTTP.

Тема 12. Программирование сетевых приложений.

Понятие сокета. Потоковые и датаграммные сокеты, сокеты домена UNIX. Реализация клиента и сервера для взаимодействия через сокеты. Создание высокопроизводительного сервера: неблокирующий ввод-вывод, функции select и epoll.

Тема 13. Файловая система.

Устройство файловых систем. Индексные дескрипторы (inodes). Сетевые диски. Диски в оперативной памяти. Механизм FUSE.

Тема 14. Ядро Linux.

Общая характеристика и структура. Модули ядра. Создание простейшего модуля. Конфигурирование, сборка и установка «ванильного» ядра Linux.

Тема 15. Виртуализация.

Механизм chroot, использование и обход «защиты». Механизмы контейнеризации: cgroups и namespaces. Виртуализация аппаратного обеспечения.

УЧЕБНО-МЕТОДИЧЕСКАЯ КАРТА УЧЕБНОЙ ДИСЦИПЛИНЫ

Дневная форма получения образования

Номер раздела, темы	Название раздела, темы	Количество аудиторных часов				Количество часов–УСР	Форма контроля знаний
		Лекции	Практические и семинарские занятия	Лабораторные занятия	Иное		
1	2	3	4	5		6	7
1	Введение	2					Устный опрос
2	Системные вызовы	1		2			Лабораторная работа с ее устной защитой
3	Ввод-вывод	1		2			Лабораторная работа с ее устной защитой
4	Сигналы	1		2			Лабораторная работа с ее устной защитой
5	Порождение процессов	2					Устный опрос
6	Неименованные каналы	1		2			Лабораторная работа с ее устной защитой
7	Потоки	1		2			Лабораторная работа с ее устной защитой. Контрольная работа
8	Внутренняя реализация потоков	1					Устный опрос
9	Управление памятью	2		2			Лабораторная работа с ее устной защитой
10	Структура программы	1		2			Лабораторная работа с ее устной защитой. Коллоквиум
11	Сети TCP/IP	1					Устный опрос
12	Программирование сетевых приложений	1		2			Лабораторная работа с ее устной защитой. Контрольная работа
13	Файловая система	1					Устный опрос
14	Ядро Linux	2		2			Лабораторная работа с ее устной защитой
15	Виртуализация	2		2			Лабораторная работа с ее устной защитой. Тестирование
ИТОГО		**20**		**20**			

ИНФОРМАЦИОННО-МЕТОДИЧЕСКАЯ ЧАСТЬ

Перечень основной литературы

1. *Таненбаум Э.* Современные операционные системы / Э. Таненбаум, Х. Бос – 4-е изд. – СПб.: Питер, 2019. – 1120 с.
2. *Kerrisk M.* The Linux Programming Interface / Michael Kerrisk. – San Francisco: No Starch Press, Inc., 2010. – 1556 p.
3. *Керниган Б.* UNIX. Программное окружение / Б. Керниган, Р. Пайк. – М.: Символ-Плюс, 2012. – 416 с.

Перечень дополнительной литературы

1. *Далхаймер М.* Запускаем Linux / М. Далхаймер, М. Уэлш. – М.: Символ-Плюс, 2012. – 992 с.
2. *Бейер Б.* Site Reliability Engineering. Надежность и безотказность как в Google / Б. Бейер, К. Джоунс, Н. Р. Мерфи, Д. Петофф –СПб.: Питер, 2018. – 592 с.
3. *Моли Б.* Unix/Linux. Теория и практика программирования. – М.: КУДИЦ-Образ, 2004. – 576 с.
4. *Стивенс Р.* UNIX. Профессиональное программирование / Р. Стивенс, С. Раго. – 2-е изд. – СПб.: Символ-Плюс, 2007. – 1040 с.
5. *Керниган Б.* Язык программирования C = The C programming language / Б. Керниган, Д. Ритчи. – 2-е изд. – М.: Вильямс, 2009. – 304 с.
6. *Herlihy M.* The Art of Multiprocessor Programming / Maurice Herlihy, Nir Shavit. – San Francisco: Morgan Kaufmann Publishers Inc., 2012. – 536 p.

Перечень рекомендуемых средств диагностики и методика формирования итоговой оценки

Для диагностики компетенции в рамках учебной дисциплины рекомендуется использовать следующие формы:

1. Устная форма: устный опрос, коллоквиум.
2. Письменная форма: контрольные работы, тестирование.
3. Устно-письменная форма: отчеты по лабораторным работам с их устной защитой, оценивание на основе проектного метода.

Формой текущей аттестации по дисциплине «Внутреннее устройство ОС семейства UNIX» учебным планом предусмотрен зачет.

При формировании итоговой оценки используется рейтинговая оценка знаний студента, дающая возможность проследить и оценить динамику процесса достижения целей обучения. Рейтинговая оценка предусматривает использование весовых коэффициентов для текущего контроля знаний студентов по дисциплине.

Примерные весовые коэффициенты, определяющие вклад текущего контроля знаний и текущей аттестации в рейтинговую оценку (формирование оценки за текущую успеваемость):

- отчёты по лабораторным работам с их устной защитой – 30 %;
- контрольные работы – 15 %;
- коллоквиум – 10 %;
- устный опрос – 30%;
- тестирование – 15%.

Рейтинговая оценка по дисциплине рассчитывается на основе оценки текущей успеваемости и зачетной оценки с учетом их весовых коэффициентов Вес оценка по текущей успеваемости составляет 40 %, зачетная оценка – 60%.

Примерная тематика лабораторных занятий

Занятие № 1. *Лабораторная работа «Системные вызовы».*
Занятие № 2. *Лабораторная работа «Ввод-вывод».*
Занятие № 3. *Лабораторная работа «Сигналы».*
Занятие № 4. *Лабораторная работа «Процессы».*
Занятие № 5. *Лабораторная работа «Потоки».*
Занятие № 6. *Лабораторная работа «Виртуальная память».*
Занятие № 7. *Лабораторная работа «Программы и библиотеки».*
Занятие № 8. *Лабораторная работа «Разработка сервера».*
Занятие № 9. *Лабораторная работа «Сборка ядра Linux».*
Занятие № 10. *Лабораторная работа «Контейнеризация».*

Описание инновационных подходов и методов к преподаванию учебной дисциплины (эвристический, проективный, практико-ориентированный)

При организации образовательного процесса большинства практических занятий используется практико-ориентированный подход, который предполагает освоение содержания учебного материала через решение практических задач, а также приобретение навыков эффективного выполнения разных видов профессиональной деятельности.

Кроме этого, при организации образовательного процесса используется комбинация методов группового обучения, проектного обучения и учебной дискуссии. Комбинация методов предполагает: ориентацию на генерирование идей, приобретение навыков для решения исследовательских, творческих и коммуникационных задач, появление нового уровня понимания изучаемой темы, применение знаний (теорий, концепций) при решении проблем, определение способов их решения.

Методические рекомендации по организации самостоятельной работы обучающихся, подготовка к зачёту

Для организации самостоятельной работы студентов магистратуры по учебной дисциплине следует использовать современные информационные технологии: систему автоматического тестирования (образовательный портал) iRunner (https://acm.bsu.by/) и систему AnyTask (https://anytask.org/school/bsu) – инструменты с эффективной функциональностью контроля, тренинга и самостоятельной работы. Рекомендуется разместить в сетевом доступе комплекс учебных и учебно-методических материалов (учебно-программные материалы, учебное издание для теоретического изучения дисциплины, презентации лекций, методические указания к практическим занятиям, электронные версии домашних заданий, материалы текущего контроля и текущей аттестации, позволяющие определить соответствие учебной деятельности обучающихся требованиям образовательных стандартов высшего образования и учебно-программной документации, в том числе вопросы для подготовки к экзамену, задания, вопросы для самоконтроля, список рекомендуемой литературы, информационных ресурсов и др.).

Примерный перечень заданий к зачёту

1. infinite-loop

Напишите на языке C программу, в которой выполняется бесконечный цикл. В процессе работы программа не должна ничего выводить в стандартные потоки. При нажатии клавиш Ctrl+C программа должна завершаться и печатать число итераций, которые она успела выполнить, в формате `Iterations: XXX`.

2. fork

Напишите на языке C программу, которая иллюстрирует создание нового процесса в UNIX-системе.

- Родительский процесс выводит на экран свой PID.
- Родительский процесс создаёт дочерний процесс с помощью системного вызова fork().
- Родительский процесс выводит на экран PID нового процесса.
- Родительский процесс входит в режим ожидания завершения дочернего процесса.
- Дочерний процесс выводит на экран свой PID.
- Дочерний процесс спит в течение одной секунды.
- Дочерний процесс завершается с кодом 42.
- Родительский процесс получает код выхода дочернего процесса и выводит его на экран.

3. fork и vfork

Необходимо сравнить производительность системных вызовов fork() и vfork(). Требуется выполнить многократный запуск утилиты /bin/true (или какой-либо другой на усмотрение студента) с ожиданием её завершения, измерить общее время. Определить, сколько времени занимает один запуск программы в обоих случаях.

4. xargs

Напишите простейший вариант утилиты `xargs`. В качестве аргументов командной строки указывается некоторая программа (возможно, с дополнительными аргументами). На стандартный ввод поступает *n* строк. Ваш `xargs` должен запустить команду последовательно *n* раз, передавая команде строку из ввода как аргумент командной строки (последний). Для простоты можно полагать, что все строки на входе имеют длину не более 1000 байт. Можно не проверять статусы завершения процессов.

5. pipeline

Необходимо реализовать на языке C программу, которая выполняла бы роль конвейера (pipeline) из командной оболочки. Программа должна запускать новые дочерние процессы с указанными аргументами и связывать их между собой, осуществляя перенаправление ввода-вывода: то, что выводит на поток стандартного вывода предыдущий процесс, попадает в поток стандартного ввода следующего процесса.

6. cat

Разработайте на языке C свою собственную версию утилиты `cat`. Если программа запущена без параметров, то она должна просто копировать все данные с *stdin* на *stdout* без изменений. Если задан один или несколько аргументов, то все они трактуются как пути к файлам, данные из которых нужно прочитать и последовательно вывести на *stdout*. В случае, если указан несуществующий файл или в процессе чтения файла возникла ошибка, выведите сообщение об ошибке на *stderr* (в произвольной форме) и продолжите обработку следующих аргументов. Если всё прошло успешно, код выхода должен быть равен нулю, иначе — единице.

7. abcabcabc

С помощью библиотеки POSIX Threads реализуйте на языке C следующую демо-программу. Три потока одновременно пишут на стандартный вывод символы: первый поток — букву *a*, второй поток — букву *b*, третий — букву *c*. Каждый поток должен вывести ровно 10000 букв и завершиться, главный поток должен дождаться завершения рабочих потоков.

8. 012345

Даётся некорректный код на языке C, в котором реализован вывод чисел из разных потоков. При помощи примитивов синхронизации организуйте

попеременную работу двух потоков так, чтобы в результате получилась последовательность 0, 1, 2, 3, 4, 5, ...

9. 10 TiB

Напишите программу на C, которая будет выделять 10 ТиБ виртуальной памяти (2^{40} байт). Проверка будет осуществляться на ОС Linux x86-64. Программа должна выделить память и ожидать ввода от пользователя, чтобы в этот момент можно было проверить объём памяти VIRT при помощи `htop` или другого инструмента. После нажатия клавиши ввода память должна корректно освобождаться, а процесс должен завершаться.

10. extsort

Отсортируйте файл с целыми числами во внешней памяти. Для ввода-вывода требуется использовать механизм отображения файлов в память (memory mapping). На входе бинарный файл *data.bin* содержит 64-битные беззнаковые целые числа, записанные с использованием порядка байт little-endian (как это принято на архитектуре x86). Количество чисел явно не указывается (его можно определить исходя из размера файла). Данные сортируются на месте (in-place).

11. Guess the Number

Нужно написать программу (клиент), которая будет играть с предоставленной программой (сервером) в игру «Угадай число». Межпроцессное взаимодействие осуществляется посредством сокета домена UNIX. В качестве дополнительного задания предлагается также реализовать аналогичную программу, работающую по протоколу TCP.

12. /dev/nulll

Требуется реализовать внешний модуль для ядра Linux, который создаёт новое символьное устройство под названием /dev/nulll (три латинские буквы L — от «/dev/null Limited Edition»). По смыслу оно должно работать так же, как стандартное нулевое устройство. Но с двумя дополнительными возможностями: должен осуществляться учёт размера записанных данных и должно поддерживаться ограничение «места на диске».

Рекомендуемая тематика контрольных работ

Контрольная работа №1. Ввод-вывод. Процессы и потоки.

Контрольная работа №2. Управление памятью. Работа с сетью.

Коллоквиум. Основы системного программирования под UNIX.

Текущий контроль знаний проводится в соответствии с учебно-методической картой дисциплины.

ПРОТОКОЛ СОГЛАСОВАНИЯ УЧЕБНОЙ ПРОГРАММЫ УВО

Название учебной дисциплины, с которой требуется согласование	Название кафедры	Предложения об изменениях в содержании учебной программы учреждения высшего образования по учебной дисциплине	Решение, принятое кафедрой, разработавшей учебную программу (с указанием даты и номера протокола)
Проектирование и реализация языков программирования	Дискретной математики и алгоритмики	Нет	Оставить содержание учебной дисциплины без изменения, протокол № 9 от 26 декабря 2019 г.
Технологии проектирования и разработки высоконагруженных веб-систем	Дискретной математики и алгоритмики	Нет	Оставить содержание учебной дисциплины без изменения, протокол № 9 от 26 декабря 2019 г.

ДОПОЛНЕНИЯ И ИЗМЕНЕНИЯ К УЧЕБНОЙ ПРОГРАММЕ

на _____/_____ учебный год

№№ пп	Дополнения и изменения	Основание

Учебная программа пересмотрена и одобрена на заседании кафедры дискретной математики и алгоритмики (протокол № ____ от _____ 20__ г.)

Заведующий кафедрой

_____ _____ _____

(ученая степень, звание) (подпись) (И.О. Фамилия)

УТВЕРЖДАЮ

Декан факультета

_____ _____ _____

(ученая степень, звание) (подпись) (И.О. Фамилия)