

УТВЕРЖДАЮ  
Проректор по учебной работе  
Ю. А. Самарский  
10 июня 2010 г

ПРОГРАММА  
по дисциплине ИНФОРМАТИКА И ПРИМЕНЕНИЕ  
КОМПЬЮТЕРОВ В НАУЧНЫХ ИССЛЕДОВАНИЯХ  
(системное программное обеспечение). Продвинутый курс.

по направлению подготовки:

010900 «Прикладные математика и физика»

факультеты: ФРТК, ФОПФ, ФАКИ, ФМБФ, ФФКЭ,  
ФАЛТ, ФУПМ

кафедра информатики

курс II

семестр 3\_

лекции - 34 (час.)

лабораторные

занятия - 68 (час.)

Экзамен - нет

Зачет дифф. - 3 сем.

Два задания

Две контрольные работы

ВСЕГО АУДИТОРНЫХ ЧАСОВ - 102

Программу составили: академик РАН В. П. Иванников,  
доцент, к.ф.-м.н. В. Е. Карпов,  
доцент, к.ф.-м.н. К. А. Коньков

Программа обсуждена на заседании кафедры информатики 21 мая  
2010 г.

Заведующий кафедрой  
д.ф.м.н., профессор

И. Б. Петров

Компетенции обучающегося, формируемые в результате освоения

дисциплины:

ОК-10, ОК-11, ОК-12, ОК-13, ПК-6, ПК-12, ПК-14. Структура преподавания дисциплины.

**1. Введение.** Цели и задачи курса. Понятие о вычислительном комплексе. Системное программное обеспечение и операционные системы. Краткая история эволюции вычислительных систем. Взаимное влияние *software* и *hardware*. Автономные, сетевые и распределенные операционные системы. Классификация автономных операционных систем по их назначению и структуре.

Знакомство с операционной системой *UNIX*. Системные вызовы и библиотека *libc*. Понятия *login* и *password*. Упрощенное устройство файловой системы в *UNIX*. Полные имена файлов. Текущая директория. Относительные имена файлов. Домашняя директория пользователя. Команда *man* - универсальный справочник. Команды *cd* и *ls*. Перенаправление стандартного ввода и стандартного вывода. Простейшие команды работы с файлами - *cat*, *cp*, *mkdir*, *mv*, *rm*. Шаблоны имен файлов. Пользователь и группа. Системные вызовы *getuid()* и *getgid()*. Команды *chown* и *chgrp*. Права доступа к регулярному файлу и к директории. Команда *chmod*. Маска создания файлов. Команда *umask*. Редактирование файлов, компиляция и запуск программ.

**2. Процессы и их планирование в операционной системе.** Понятие процесса. Процесс и программа. Состояния процесса. Управляющий блок процесса и его контекст. Операции над процессами. Переключение контекста. Уровни планирования процессов. Критерии планирования и требования к алгоритмам планирования. Параметры планирования. Вытесняющее и невытесняющее планирование. Алгоритмы планирования: *FCFS*, *RR*, *SJF*, гарантированное планирование, приоритетное планирование, многоуровневые очереди, многоуровневые очереди с обратной связью.

Понятие процесса в *UNIX*, его контекст. Идентификация процесса. Краткая диаграмма состояний процессов в *UNIX*.

Иерархия процессов. Системные вызовы *getpid()* и *getppid()*. Создание процесса в *UNIX*. Системный вызов *fork()*. Завершение

процесса. Функция *exit()*. Параметры функции *main()* в языке C. Переменные среды и аргументы командной строки. Изменение пользовательского контекста процесса. Семейство функций для системного вызова *exec()*.

**3. Кооперация процессов.** Взаимодействующие и независимые процессы. Категории средств связи. Установление и завершение связи. Прямая и косвенная адресация. Информационная валентность процессов и средств коммуникации. Симплексная, дуплексная и полудуплексная связь. Потоки ввода-вывода и сообщения. Буферизация данных. Надежность обмена информацией. Нити исполнения и их отличие от процессов. *Interleaving*, *race condition* и взаимоисключения. Условия Бернштейна. Понятие критической секции процесса. Программные алгоритмы организации взаимодействия процессов и предъявляемые к ним требования. Аппаратная поддержка взаимоисключений. Семафоры, мониторы Хора и сообщения. Доказательство эквивалентности мониторов, семафоров и сообщений.

Понятие потока ввода-вывода в операционной системе *UNIX*. Работа с файлами через системные вызовы и через функции стандартной библиотеки. Файловый дескриптор. Наследование файловых дескрипторов при системных вызовах *fork()* и *exec()*. Системные вызовы *open()*, *read()*, *write()*, *close()*. *FIFO* и *pipe*. Системные вызовы *pipe()*, *mknod()*, функция *mkfifo()*. Особенности системных потоковых вызовов при работе с *FIFO* и *pipe*. Преимущества и недостатки потокового обмена данными. *IPC* в *UNIX*. Пространство имен. Адресация в *System V IPC*. Функция *ftok()*. Дескрипторы *System V IPC*. Разделяемая память. Системные вызовы *shmget()*, *shmat()*, *shmdt()*, *shmctl()*. Команды *ipcs* и *ipcrm*. Нить исполнения (*thread*) в *UNIX*, ее идентификатор. Функция *pthread\_self()*. Создание и завершение нити исполнения. Функции *pthread\_create()*, *pthread\_exit()*, *pthread\_join()*. Семафоры в *UNIX*. Отличие операций над *UNIX* семафорами от классических операций. Системные вызовы *semget()*, *semop()*,

*semctl()*. Понятие о *POSIX* семафорах. Очереди сообщений в *UNIX*. Системные вызовы *msgget()*, *msgsnd()*, *msgrcv()*, *msgctl()*. Понятие мультиплексирования. Мультиплексирование сообщений. Модель взаимодействия процессов клиент-сервер. Неравноправность клиента и сервера.

4. **Управление памятью.** Связывание адресов. Простейшие схемы управления памятью: схема с фиксированными разделами, своппинг, схема с переменными разделами. Проблема размещения больших программ. Понятие виртуальной памяти. Страничная память. Сегментная и сегментно-страничная организации памяти. Таблица страниц. Ассоциативная память. Иерархия памяти. Размер страницы. Исключительные ситуации при работе с памятью. Стратегии управления страничной памятью: выборки, размещения и замещения страниц. Алгоритмы замещения страниц: *FIFO*, *OPT*, *LRU* и другие. *Thrashing*. Свойство локальности. Модель рабочего множества. Аппаратно-независимая модель памяти процесса.

5. **Файловые системы.** Имена, структура, типы и атрибуты файлов. Операции над файлами. Директории. Операции над директориями. Защита файлов. Интерфейс файловой системы и ее общая структура. Методы выделения дискового пространства: непрерывная последовательность блоков, связный список, связный список с индексацией, индексные узлы. Управление свободным и занятым дисковым пространством: битовый вектор, связный список. Размер блока на диске. Реализация директорий. Монтирование файловых систем. Надежность и целостность файловых систем.

Разделы носителя информации (*partitions*) в *UNIX*. Логическая структура файловой системы и типы файлов в *UNIX*. Организация файла на диске в *UNIX* на примере файловой системы *s5fs*. Понятие индексного узла (*inode*). Организация директорий (каталогов) в *UNIX*. Понятие суперблока. Указатель текущей позиции в файле. Системная таблица файлов и таблица индексных узлов открытых файлов. Операции над файлами и директориями. Понятие жестких и мягких связей. Системные вызовы и команды для выполнения операций над файлами и

директориями: *chmod*, *chown*, *chgrp*, *open()*, *creat()*, *read()*, *write()*, *close()*, *stat()*, *fstat()*, *istat()*, *ftruncate()*, *iseek()*, *Unk()*, *symlink()*, *unlink()*. Функции для изучения содержимого директорий (*opendir()*, *readdir()*, *rewinddir()*, *closedir()*). Понятие о файлах, отображаемых в память (*memory mapped* файлах). Системные вызовы *mmap()*, *mmap()*. Понятие виртуальной файловой системы. Монтирование файловых систем в *UNIX*.

**6. Система управления вводом-выводом.** Общие сведения об архитектуре компьютера. Структура контроллера устройства. Опрос устройств и прерывания. Исключительные ситуации и системные вызовы. Прямой доступ к памяти (*Direct Memory Access - DMA*). Структура системы ввода-вывода. Систематизация внешних устройств и интерфейс между базовой подсистемой ввода-вывода и драйверами. Функции базовой подсистемы ввода-вывода. Блокирующиеся, не блокирующиеся и асинхронные системные вызовы. Буферизация и кэширование. *Spooling* и захват устройств. Обработка прерываний и ошибок. Планирование запросов. Алгоритмы планирования запросов к жесткому диску: *FCFS*, *SSTF*, *SCAN*, *C-SCAN*, *LOOK*, *C-LOOK*. Блочные и символьные устройства в *UNIX*. Понятие драйвера. Блочные, символьные драйверы, драйверы низкого уровня. Файловый интерфейс к драйверам. Коммутатор устройств. Старший и младший номер устройства. Понятие сигнала в *UNIX*. Способы возникновения сигналов и виды их обработки. Понятия группы процессов, сеанса, лидера группы, лидера сеанса, управляющего терминала сеанса, текущей и фоновой групп процессов. Системные вызовы *getpgrp()*, *setpgrp()*, *getpgid()*, *setpgid()*, *getsid()*, *setsid()*. Системный вызов *kill()* и команда *kill()*. Особенности получения терминальных сигналов текущей и фоновой группой процессов. Получение сигнала *SIGHUP* процессами при завершении лидера сеанса. Системный вызов *signal()*. Установка собственного обработчика сигнала. Сигналы *SIGUSR1* и *SIGUSR2*. Использование сигналов для синхронизации процессов. Завершение порожденного процесса. Системный вызов *waitpid()* Сигнал *SIGCHLD* и его

игнорирование. Возникновение сигнала. *SIGPIPE* при попытке записи в *pipe* или *FIFO*, который никто не собирается читать. Понятие о надежности сигналов *POSIX функции* для работы с сигналами.

**7. Сети и сетевые операционные системы.** Причины объединения компьютеров в сети. Сетевые и распределенные операционные системы. Взаимодействие удаленных процессов как основа работы вычислительных сетей. Локальные и глобальные вычислительные сети. Топология компьютерных сетей. Взаимная синхронизация вычислительных комплексов: обнаружение коллизий (*CSMA/CD*), метод передачи эстафетной палочки (*token passing*), использование слотов для данных. Удаленная адресация и разрешение адресов. Понятие о *DNS*. Локальная адресация. Понятие порта. Полные адреса. Понятие сокета (*socket*). Фиксированная, виртуальная и динамическая маршрутизация. Сети, коммутируемые цепями, сообщениями и пакетами данных. Связь с установлением логического соединения и передача данных с помощью сообщения. Многоуровневая модель построения сетевых вычислительных систем. Семейства и стеки протоколов. Эталонная модель *OSI/ISO*.

Краткая история семейства протоколов *TCP/IP*. Общие сведения об архитектуре семейства протоколов *TCP/IP*. Уровень сетевого интерфейса. Уровень *Internet*. Протоколы *IP, ICMP, ARP, RARP*. *Internet*-адреса. Транспортный уровень. Протоколы *TCP* и *LIDP*. Понятие порта. Понятие *encapsulation*. Уровень приложений/процессов. Использование модели клиент-сервер для взаимодействия удаленных процессов. Понятие *socket* в *UNIX*. Организация связи между удаленными процессами с помощью датаграмм Организация связи между процессами с помощью установки логического соединения Сетевой порядок байт. ФУНКЦИИ *htons(), htonl(), ntohs(), ntohl()*. Функции преобразования IP-адресов *inet\_ntoa(), inet\_aton()*. Функция *bzero()*. Системные вызовы *socket(), bind(), sendto(), recvfrom(), accept(), listen(), connectQ*.

**8. Проблемы безопасности операционных систем.** Классификация угроз. Формализация подхода к обеспечению информационной безопасности. Классы безопасности. Политика безопасности. Криптография как одна из базовых технологий безопасности ОС. Шифрование с симметричными и асимметричными ключами. Правило Кирхгофа. Алгоритм *RSA*. Идентификация и аутентификация. Пароли, уязвимость паролей. Авторизация. Разграничение доступа к объектам ОС. Домены безопасности. Матрица доступа. Недопустимость повторного использования объектов. Аудит, учет использования системы защиты.

## **Учебно-методическое и информационное обеспечение дисциплины**

### **Основная литература**

1. *Карпов В. Е., Коньков К. А.* Основы операционных систем. - М.: ИНТУИТ.РУ «Интернет-Университет Информационных технологий», 2005.

### **Дополнительная литература**

1. *Стопинге В.* Операционные системы. - М.: Вильямс, 2001.
2. *Олифер В.Г., Олифер Н.А.* Сетевые операционные системы. СПб.: Питер, 2001.
3. *Стивене У.* *UNIX*: Взаимодействие процессов. СПб.: Питер, 2002.
4. *Стивене У.* *UNIX*: Разработка сетевых приложений. СПб.: Питер, 2003.
5. *Таненбаум Э.* Современные операционные системы. СПб.: Питер, 2002.
6. *Дейтел Х.М., Дейтел П.Дж., Хоффнес Д.Р.* Операционные системы. Основные принципы: 3-е издание. - М.: ООО «Бином-Пресс», 2006.
7. *Дейтел Х.М., Дейтел П. Дж., Хоффнес Д. Р.* Операционные системы. Распределенные системы, сети, безопасность: 3-е издание. - М.: ООО «Бином-Пресс», 2006.

8. *Silberschatz A., Galvin P. B. Operating System Concepts. 6<sup>th</sup> edition.*  
- John Willey & Sons, 2002.

### Электронные ресурсы

1. <http://cs.mipt.ru>
2. <http://acm.mipt.ru>

### Задание 1

{срок сдачи 25-30 октября}

1. Напишите программу *useless* (*UNIX SYSTEM EXTREMELY LATE EXECUTION SOFTWARE SYSTEM*), которая читает файл и запускает указанные в нем программы с указанной задержкой от времени старта программы *useless*.  
Формат записи в файле:

<время задержки в секундах> <программа для выполнения>  
Файл не является упорядоченным по временам задержки.

2. Для того чтобы не допустить потерю информации при порче диска, обычно используют резервное копирование файлов (*backup*). Простейшей формой (*backup*) является копирование всех файлов из одной директории в другую. Этот способ требует много времени и места на диске. Напишите программу, осуществляющую более интеллектуальный подход.

Программа должна брать из командной строки два параметра: исходную директорию и директорию назначения. Она должна рекурсивно сканировать исходную директорию, делать копии всех файлов, для которых ранее не делались копии или которые были изменены с момента последнего *backup*'а, размещая их в соответствующих местах директории назначения.

После копирования каждого файла должна вызываться команда сжатия *gzip*. Это уменьшит требуемый размер дисковой памяти; а файл будет переименован с добавлением расширения *.gz*. Все возникающие ошибки (нет исходной директории, файл закрыт для чтения и т.д.) должны корректно обрабатываться с выдачей соответствующего сообщения.

### Задание 2

(срок сдачи 13-18 декабря)



1. Напишите программу *runsim*, осуществляющую контроль количества одновременно работающих *UNIX*-приложений. Программа читает *UNIX-команду* со стандартного ввода и запускает ее на выполнение. Количество одновременно работающих команд не должно превышать  $N$ . где  $N$  - параметр командной строки при запуске *runsim*. При попытке запустить более чем  $N$  приложений выдайте сообщение об ошибке. Программа *runsim* должна прекращать свою работу при возникновении признака конца файла на стандартном вводе.

2. На мойке посуды в ресторане работают два человека. Один из них моет посуду, второй вытирает уже вымытую. Времена выполнения операций мытья и вытирания посуды меняются в зависимости от того, что моется. Стол для вымытой, но не вытертой посуды имеет ограниченные размеры.

Смоделируйте процесс работы персонала следующим образом: каждому работнику соответствует свой процесс. Времена выполнения операций содержатся в двух файлах. Каждый файл имеет формат записей:

<тип посуды> : <время операции>

Стол вмещает  $N$  предметов независимо от их наименования. Значение  $N$  задается как параметр среды *TABLE LIMIT* перед стартом процессов. Грязная посуда, поступающая на мойку, описывается файлом с форматом записи:

<тип посуды> : <количество предметов> Записи с одинаковым типом посуды могут встречаться неоднократно. Организуйте передачу посуды от процесса к процессу: а) через файл, используя семафоры для синхронизации;

б) через *pipe*;

в) через сообщения;

г) через разделяемую память;

д) через *sockets*.

Усл. печ. л. 0,75. Тираж 810 экз.