«Национальный открытый институт»

Сибирев В.Н., Спиридонов В.В.

Операционные системы

Методические указания к выполнению практических работ

Рекомендовано Методической комиссией по качеству
Национального открытого института
для студентов, обучающихся по направлению
09.03.03 «Прикладная информатика»

Санкт-Петербург
2016

УДК 519.2.06 (07)
ББК 32.973
С34

Настоящая методическая разработка соответствует требованиям федерального государственного образовательного стандарта высшего профессионального образования.

В методических указаниях приводятся пять практических работы, в которых рассматриваются основные действия с операционными системами Windows и Linux (Unix), которые необходимо освоить бакалавру по информационным технологиям. В данном комплексе практических работ рассматривается использование только лишь некоторой их части.

УДК 519.2.06 (07)
ББК 32.973

Введение

Операционные системы (ОС) являются неотъемлемой частью современных компьютеров и вычислительных систем. Целью данного комплекса практических работ является ознакомление студентов с основными операциями, выполняемыми в рамках настройки и использования операционных систем.

В ОС имеется большое количество функции, предназначенных для выполнения служебных и настроечных операций в системе. Для обычного пользователя, часть из них выводится средствами стандартного (обычно, оконного) интерфейса. Однако специалист по информационным технологиям должен уметь использовать и более тонкие настройки ОС и соответствующие их средства. Информацию о подробном использовании всех функций ОС, обычно можно получить по типовым подсказкам системы или на соответствующих сайтах.

Библиографический список:

1. Гордеев А.В Операционные системы: Учебник для вузов. 2-е изд.- СПб.: Питер, 2009..

2. Таненбаум Э. Современные операционные системы. 3-е изд. – СПб. Питер, 2010.

б) дополнительная литература

3. Назаров С.В., Широков А.И. Современные операционные системы. - М.: Бином, 2013.

4. Олифер В.Г., Олифер Н.А. Сетевые операционные системы: Учебник для вузов. 2-е изд. - СПб.: Питер, 2009.

5. Руссинович М., Соломон Д. Внутреннее устройство Microsoft Windows. 6-е изд. — СПб.: Питер, 2013. — 800 с.: ил. — (Серия «Мастер-класс»).

6. Архитектура ЭВМ и систем: учебно-методический комплекс / Национальный минерально-сырьевой университет «Горный». Сост.: М.В. Копейкин, В.В. Спиридонов, Е.О. Шумова. – СПб, 2013, 126 с.

7. Операционные системы: учебно-методический комплекс / Сост.: Б.М. Илюшкин, – СПб, СЗТУ, 2010, 155 с.

8. Джонсон, М., Троан, Э. Разработка приложений в среде Linux/Пер. с англ., 2-е изд. − М.: ООО «И.Д. Вильямс», 2007..

Практическая работа № 1

Установка ОС в виртуальных машинах

1. Цель работы

Целью практической работы является ознакомление с механизмом, реализующим принцип виртуализации применительно к операционным системам, и получение практических навыков установки виртуальных машин и виртуальных ОС в них.

2. Общие сведения

Идея виртуальной машины явилась развитием концепции виртуальной памяти.

Аналогично последней, виртуальная машина предоставляет пользователю возможность работать в программной среде, эмулирующей поведение реальной ЭВМ и позволяющей производить на ней установку различных операционных систем, функционирующих независимо друг от друга. (Существуют и несколько отличные варианты применения термина виртуальная машина.)

Виртуализацию операционных систем часто используют для организации многопользовательских режимов работы на ЭВМ.

Другими частыми применениями виртуальных машин является постановка различных экспериментов на ЭВМ или использование имеющегося программного обеспечения, ориентированного на работу в некоторой ОС, отсутствующей на имеющейся ЭВМ при невозможности или недопустимости ее установки.

Одной из главных функций, реализуемой виртуальными ЭВМ, является повышения надежности и безопасности работы подсистем информационной системы, установленной на ЭВМ, за счет изолированность их друг от друга.

Впрочем, изоляция подсистем может быть и аппаратной.

Исторически реализация виртуальных систем имела три основных технологии:

- среда языка программирования
- монитор виртуальных машин (гипервизор), имеющий, в свою очередь, реализации с привлечением специальных аппаратных средств, на основе определенной ОС (хост-ОС), в комбинированных вариантах
- специальные приложения – эмуляторы.

Наиболее распространенными автономными виртуальными машинами являются

1. VirtualBox (Oracle) (https://www.virtualbox.org/)
2. VMware Workstation (http://www.vmware.com/)
3. Virtual PC (Microsoft)

VirtualBox - бесплатная программа, имеющая русскоязычный интерфейс. Создана в 2007 г. компанией InnoTek в вариантах с открытым и закрытым исходными кодами, бесплатных при условии некоммерческого использования. В 2008 г. перекуплена компанией Sun Microsystems, которая, в свою очередь, в 2010 г. была поглощена компанией Oracle.

Представляет собой систему виртуализации, работающую в Windows, Linux и Mac OS и обеспечивает использование в качестве гостевых операционных систем Windows (2000/XP/2003/Vista/Seven), Linux (Ubuntu/Debian/ OpenSUSE/ Mandriva и пр.), OpenBSD, FreeBSD, OS/2 Warp.

VirtualBox поддерживает многопроцессорность и многоядерность, виртуализацию аудиоустройств и сетевых устройств при различных вид ах сетевого взаимодействия (NAT, Host Network, Bridge, Internal), средства USB-host. Отличается невысоким потреблением ресурсов ПК, возможностью сохранения снимков виртуальной машины (snapshots), а также имеет интерфейс командной строки.

Программа имеет также пакет VirtualBox Guest Additions, который может устанавливаться в гостевую операционную систему, расширяя возможности её взаимодействия с системой виртуализации и хост-системой.

3. Порядок выполнения практической работы

1. Установить виртуальную машину VM VirtualBox
2. Установить в виртуальной машине операционную систему Linux
3. Установить в виртуальной машине операционную систему Windows.
4. Установить интерпретатор команд командной оболочки Linux для Windows – CygWin.
5. Создать резервные копии установленных машин.

Виртуальные машины должны иметь название - "FIO_OS". Пользователь ОС должен иметь имя "FIO"

Установленные ОС должны позволять использовать:

- CDROM
- сеть
- USB накопители,

а также обеспечивать возможность обмена с host-машиной файлами через общие папки.

4. Указания к выполнению практической работы

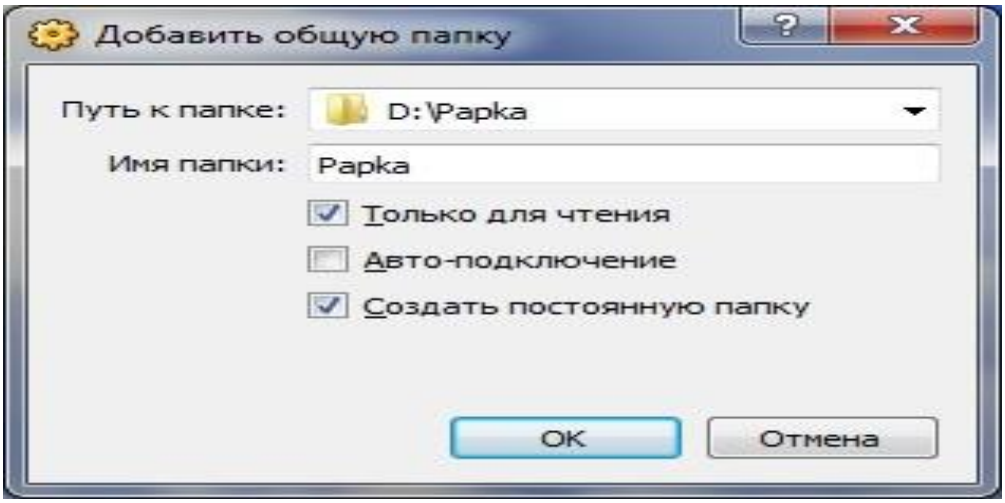
При выполнении п.5 порядка работы для организации доступа из гостевой ОС к файлам, хранящимся на жестком диске машины-хоста и наоборот, следует создать общие папки, содержимое которых будет видно как гостевой, так и хост машинам.

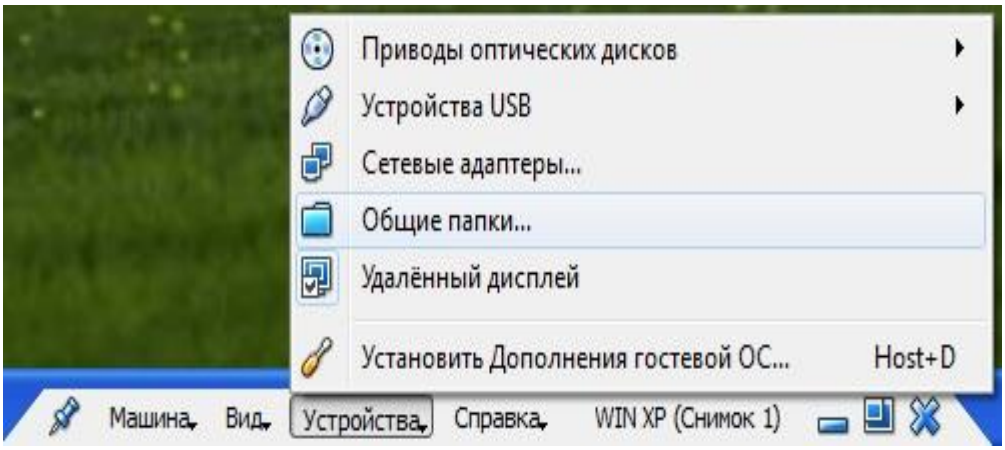
Для создания общей папки необходимо проделать следующие операции (при этом должны быть установлены дополнения гостевой ОС).

Находясь в виртуальной ОС, требуется выбрать в меню VirtualBox путь: Устройства\Общие папки…

Затем нужно добавить папку, нажав на значок «папка с плюсиком» (или кнопку Insert), и указать путь к папке, имя папки будет указанно автоматически (его можно изменить). Именно это имя в дальнейшем будет использоваться для подключения общей папки. Кроме того следует поставить галочку «Создать постоянную папку».

После создания общей папки, гостевую ОС необходимо перезагрузить. Подключенную папку можно найти в сетевом окружении:





Сетевое окружение\Вся сеть\VirtualBox Shared Folders\Vboxsvr\

Для удобства, можно создать ярлык на рабочем столе или подключить сетевой диск, тогда он будет отображаться в «Моем компьютере».

- Мой компьютер\Сервис\Подключить сетевой диск… (для Windows XP).
- Мой компьютер\Подключить сетевой диск (для Windows7 и выше).

выбрав любую букву диска, в поле «Папка» ввести: \\Vboxsvr\имя папки и поставить галочку «Восстанавливать при входе в систему».

Для доступа к общей папке в Linux необходимо ее смонтировать командой

mount -t vboxfs <имя_общей папки> <точка_монтирования>

VirtualBox общие папки Ubuntu

1) У вас должны быть установлены дополнения к гостевой ОС. Если они у вас еще не установлены, установите их, и только потом переходите к следующим пунктам. О том, как установить эти дополнения, можно прочесть в статье «VirtualBox дополнения гостевой ОС» (ссылка находится в начале данной статьи, в пункте первом).

2) Добавьте общую папку, через меню VirtualBox (о том, как это сделать см. выше, пункт второй). Есть только одно уточнение, при добавлении папки, в пункте «Имя папки», имя должно быть прописано только английскими буквами (запомните это имя).

3) Запустите терминал, и введите следующую команду:

```
cat /etc/group | grep plugdev
```

(команда grep – по сути своей – это поиск по шаблону, происходит от g/re/p (global/regular expression/print) – команды одного из первых редакторов Unix - ed)

В терминале появится вот такая строка plugdev:x:46:. Нам нужно получившееся в результате выполнения команды число (46) (скорее всего, оно будет таким же).

Открываем на редактирование файл fstab, пишем в терминале:

```
sudo gedit /etc/fstab
```

После ввода по запросу пароля, следует внести в fstab следующую запись:

papka /media/papka vboxsf defaults,gid=46 0 0

Можно добавить эту строчку в самый конец файла, внеся в нее следующие изменения:

- Вместо papka — написать имя своей общей папки (полученное при выполнении второго пункта).
- gid=46 — число, которое мы узнали ранее (если у вас оно другое, измените на свое).

Не забудьте сохранить отредактированный документ.

Создаем каталог под названием papka, в терминале введите следующее:

```
sudo mkdir /media/papka
```

Вместо papka напишите имя своей общей папки. Теперь надо перезагрузить гостевую ОС.
Ищите созданный каталог в расположении: Файловая система/media/.

5. Отчет по практической работе

Отчет по практической работе должен содержать:
1. Описание использованных при выполнении работы пакетов (инсталляций)
2. Краткое описание системы VM VirtualBox.
3. Параметры установленных ОС.
4. Экраны с изображением рабочего стола установленных систем и основными меню.

6. Литература и ссылки на информационные ресурсы

www.citforum.ru
Виртуальная машина в wikepedia

Практическая работа № 2

Операции с файлами и дисками

1. Цель работы

Целью работы является получение практических навыков выполнения операций с файлами и дисками, а также самостоятельной работы с информацией о соответствующих командах ОС.

2. Общие сведения

Для выполнения операций с дисками и файлами используются различные команды, позволяющие манипулировать дисками, директориями и файлами: проверять их, устанавливать

атрибуты, открывать и закрывать файлы, создавать и удалять директории и т.д.

Для задания типовых действий с дисками и файлами в ОС используются специальные команды, которые могут различаться в зависимости от операционной системы. Обычно, исходя из соображений межплатформенных взаимодействий систем и приложений, часть команд являются общими для различных систем, обеспечивая определенный уровень их совместимости.

2.1. Команды POSIX для работы с процессами (должны быть во всех типах ОС):

df - выводит информацию о смонтированных дисках

2.2. Команды LINUX для выполнения действий с дисками и файлами:

Форматирование дисков и создание файловых систем

fdformat – форматирование гибкого диска
fdisk – работа с разделами диска при использовании MBR
gdisk – работа с разделами диска при использовании GPT
parted - работа с разделами диска при использовании GPT и не только
mkfs – создание файловой системы (форматирование)
mformat – создает файловую систему MS-DOS

Монтирование и размонтирование файловых систем

mount - монтирование файловых систем
umount - размонтирование файловых систем

Информация о файловых системах

df - выводит информацию о смонтированных дисках
fsck – проверка файловой системы
debugfs - восстановление и получение подробной информации файловых систем
filefrag - получить информацию о фрагментации файла
mkswap – создание раздела подкачки
swapon – активизация раздела подкачки
/etc/fstab - файл для описания подключаемых при загрузке файловых систем

Создание файлов и директориев

mkdir – создать директорий

touch *file_name* – создать файл *file_name*

cat > *file_name* – создать файл *file_name* и ввести в него данные с клавиатуры (для завершения ввода следует нажать *Ctrl+D*)

Копирование / перемещение файлов и директориев

cp *src dst* – копировать файл *src* в *dst*

mv *src dst* – переместить/переименовать файл/директорий *src* в *dst*

Сцепление файлов

cat *file_name1 file_name2* – сцепить (и вывести на экран) содержимое файлов *file_name1* и *file_name2*

cat *file_name1* **>>** *file_name2* – добавить в конец файла *file_name2* данные файла *file_name1*

Редактирование и поиск файлов

locate *file_name* – поиск всех файлов, содержащих в имени *file_name*

nano *file_name* – редактирование файла *file_name*

gedit *file_name* – редактирование файла *file_name*

vi *file_name* – редактирование файла *file_name (FreeBSD)*

ee *file_name* – редактирование файла *file_name (FreeBSD)*

Вывод файлов и директориев на экран

ls *dir_name* – вывести содержимое директория *dir_name* на экран

cat *file_name* – вывести на экран содержимое файла *file_name*

head [-n] *file_name* – вывести на экран первые *n* строк файла *file_name*

more *file_name* – постраничный вывод на экран файла *file_name*

tail [-n] *file_name* – вывести на экран последние *n* строк файла *file_name* (удобно для чтения логов)

Удаление файлов и директориев

rm *file_name* – удалить файл *file_name*

rmdir *dir_name* – удалить директорий *dir_name*

[rm –rf] – удалить директорий и все содержащиеся в нем файлы

Уточнить параметры вызова любой команды и получить о ней дополнительную информацию можно вызвав команду man <имя нужной команды>. При этом на консоль будут выведена справочная информация по нужной команде.

2.3. Команды Windows для выполнения действий с дисками и файлами:

Форматирование дисков и создание файловых систем

format - форматирование диска

assign – назначение буквы диску или пути его включения в файловую систему (монтирование)

chkdsk - вывод на экран отчета о состоянии диска

chkntfs - просмотр или задание планирования автоматической проверки системы для томов файловых систем FAT, FAT32 или NTFS при запуске компьютера.

cipher - отображение или изменение шифрования папок и файлов на томах NTFS

compact - вывод сведений или изменение уплотнения файлов и каталогов в разделах NTFS

convert - преобразование томов с файловой системой FAT и FAT32 в тома с файловой системой NTFS.

defrag - поиск и объединение фрагментированных файлов

DiskPart - программа DiskPart.exe — это работающий в текстовом режиме командный интерпретатор, который позволяет управлять объектами (дисками, разделами или томами) с помощью сценариев или команд, вводимых с командной строки.

fsutil (поддерживаются только с версии Windows 5.1) - является служебной программой командной строки, которая используется для выполнения связанных задач файловых систем FAT и NTFS. Некоторые из ее подкоманд следующие:

- **file** - поиск файла по идентификатору безопасности, запрос файла в размещенных диапазонах, установка короткого имени файла, допустимой длины данных или нулевых данных для файла.
- **fsinfo** - перечисляет все диски, запрашивает тип диска, сведения о томе, специальные сведения о томе NTFS или статистику файловой системы.

volume - размонтирование тома или отображение свободного места на диске.

label - служит для создания, изменения или удаления метки тома (т. е. имени) диска.

mountvol - служит для создания, удаления и получения списка точек подключения тома.

subst - устанавливает соответствие пути в файловой системе заданному диску.

vol - отображает метку тома диска и серийный номер, если они существуют.

Создание, перемещение и удаление файлов и директориев

mkdir, md – создать директорий

copy *src dst* – копировать файл *src* в *dst*

xcopy *src dst* – копировать файл/директорий *src* в *dst*

move *src dst* – переместить/переименовать файл/директорий *src* в *dst*

replace – **замещение файлов и директориев**

rmdir. rd – удаление директория

del *file_name* **– удаление файла**

erase *file_name* **– удаление файла**

edit *file_name* **– редактирование файла**

а также перенаправление вывода в файл: > или >> и далее – имя файла.

3. Порядок выполнения практической работы

1. Запустить ОС Linux.

2. Ознакомиться со списком команд Linux для работы с дисками и файлами и параметрами их вызова.

3. Создать в виртуальной машине новый диск размером 6 ГБ и выполнить следующее:

а) создать на этом диске 5 разделов;

б) создать на этих разделах файловые системы NTFS, FAT32, EXT3 (с журналом), XFS;

в) смонтировать все разделы в подкаталоги каталога /mnt/<первые (латинские) буквы фамилии имени и отчества>/<название файловой системы>

4. Используя команды, научиться выполнять следующие виды операций с дисками и файлами:

а) монтировать/размонтировать файловую систему, в том числе флеш-накопителя;

б) выводить информацию о смонтированных дисках (в т.ч. тип файловой системы)

в) проверять файловые системы;

г) устанавливать соответствие пути в файловой системе заданному диску

д) создавать, изменять или удалять метки тома (т. е. имени) диска

е) отображать метку тома диска и серийный номер, если они существуют

ж) преобразовывать тома с файловой системой FAT и FAT32 в тома с файловой системой NTFS

з) (необязательный) отображать или изменять шифрования папок и файлов на томах NTFS

и) создавать, перемещать / переименовывать и удалять директории

к) создавать, копировать, перемещать/переименовывать, редактировать и удалять файлы

5. Для созданных дисков и файловых систем провести анализ их характеристик:

а) сделать автоматическое монтирование при загрузке

б) на каждый из разделов, созданных в п. 3, скопировать большой файл (>1Гбайт) и получить информацию о фрагментации, составить сравнительную таблицу

в) произвести тестирование файловых систем, составить сравнительную таблицу;

6. Запустить ОС Windows

7. Ознакомиться со списком команд Windows для работы с дисками и файлами и параметрами их вызова.

8. Создать в виртуальной машине новый диск размером 4 ГБ и создать на этом диска 3 раздела.

9. Используя командную строку и утилиту DISKPART и научиться выполнять основные виды операций с дисками и файлами:

а) на первом разделе создать файловую систему FAT32, а на втором – NTFS:

б) создать каталог c:\FIO\ и сделать его виртуальным диском K:

в) проверять созданные файловые системы:

г) устанавливать соответствие пути в файловой системе заданному диску:

д) присвоить метки вида (ФИО – латинскими буквами FAT32 и NTFS) двум первым разделам:

е) изменять и удалять метки тома (т. е. имени) диска:

ж) отобразить метку тома диска и серийный номер тома, если они существуют:

з) преобразовать тома с файловой системой FAT и FAT32 в тома с файловой системой NTFS

и) создавать, перемещать / переименовывать и удалять директории

к) создавать, копировать, перемещать/переименовывать, редактировать и удалять файлы, в том числе файлы NTFS с потоками.

10. Ознакомиться с возможностями расширенного командного процессора PowerShell.

4. Указания к выполнению практической работы

При выполнении пунктов 3 и 8 порядка работы создание нового диска производится при выключенной виртуальной машине.

Для добавления нового диска следует выбрать из основного меню программы VM VirtualBox пункт *(Машина/)Настройка/Носители.* Затем в окне носителей выделить строку любого из имеющихся контроллеров (SATA, IDE) и кликнуть на иконке ”*Добавить жесткий диск*” и повторить все операции (выбор типа, задание имени, назначение объема и др.), которые выполнялись при настройке этого параметра во время установки виртуальной машины.

Далее при выполнении п.п. 3 и 8 порядок разбиения добавленного диска на разделы зависит от используемого типа таблицы разделов диска (MBR или GPT) и, в некоторой степени, от семейства используемой операционной системы: Linux (Ubuntu, Fedora и др.) или Unix (FreeBSD).

Таблица разделов, используемая в MBR (Master Boot Record – Главная Загрузочная Запись) позволяет задать 4 физических раздела на диске. Эта таблица работает с любой ОС и на всех компьютерах, включая старые. Таблица GPT (GUID Partition Table – Таблица Разделов GUID – Globally Unique Identifier) работает только на новых компьютерах, использующих UEFI вместо BIOS. Эта таблица позволяет создавать более четырех основных разделов на диске и обеспечивает размеры разделов более 2Тб.

Работа с дисками в системе Linux

При выполнении п. 3,а) программы работы для создания разделов на диске в ОС семейств Linux/Unix могут использоваться либо команда ***fdisk*** (или ***cfdisk***) для таблиц разделов в формате MBR, либо команда ***parted*** (Linux) или ***gpart*** (Unix) для таблиц разделов.

Для таблицы разделов формата MBR

Определить разбиение диска на разделы можно с помощью команды ***fdisk*** с ключем ***- l*** :

fdisk –l <путь_к_диску> или

cfdisk –P {r|t} <путь_к_диску>

Обычно жесткие диски в Linux обозначаются как sda, sdb и т.д., располагаясь в директории /dev.

Для создания нового раздела следует вызвать команду ***fdisk*** без ключей, указав путь к диску. (Для выполнения команды может потребоваться задание прав суперпользователя – префикс sudo перед командой.)

[sudo] fdisk <путь_к_диску>

При этом система высветит набор ключей-команд, из которых следует воспользоваться ключом **n** для добавления раздела в таблицу или ключом **p** – для просмотра имеющейся таблицы разделов.

После внесения изменений в таблицу разделов, например, после добавления нового раздела, следует выполнить команду

[sudo] fdisk –w <путь_к_диску>**,**

записывающую изменения в таблице на диск.

Для форматирования (запись файловой системы) нового раздела следует вызвать команду ***mkfs*****,** указав путь к форматируемому разделу диска. (Для выполнения команды может потребоваться задание прав суперпользователя – префикс sudo перед командой.)

[sudo] mkfs –t **<**тип_ФС**>** <путь_к_диску>

Создать файловую систему NTFS можно и просто с помощью команды ***mkntfs*** <путь_к_диску>.

Для таблицы разделов формата GPT

В этом случае для работы с разделами жесткого диска в Linux используется команда ***parted***. Получить информацию по параметрам вызова команды можно, как обычно, вызвав команду без параметров или с ключем ***help*****.**

Просмотр имеющихся разделов на диске обеспечивает подкоманда ***print***, которую можно вызвать уже после вызова ***parted*** или сразу:

[sudo] parted <путь_к_диску> ***print***

Добавить новый раздел позволяет подкоманда ***mkpart*** или ***mkpartfs***, первая из которых создает раздел в диалоге, если не указаны требуемые значения, запрашивая имя раздела, тип файловой системы, начало и конец раздела (в мегабайтах), а вторая – раздел с файловой системой на нем. Предпочтительным считается использование первой подкоманды:

[sudo] parted <путь_к_диску> ***mkpart [fs]*** <start> **<**end**>.**

Создать на имеющемся разделе файловую систему позволяет подкоманда ***mkfs*****,** либо отдельная команда ***mkfs*** (что предпочтительнее)

[sudo] mkfs –t <тип_ФС> <путь_к_диску>.

Работа с дисками в системе Unix (FreeBSD)

Определить разбиение диска на разделы можно с помощью команды ***fdisk*** :

fdisk <путь_к_диску>

Обычно жесткие диски в Unix обозначаются как ada1, ada2 и т.д., располагаясь в директории /dev.

Однако если таблица разделов диска имеет формат GPT, то будет показан всего один раздел, охватывающий целый диск. Поэтому лучше использовать команду ***gpart*** с подкомандой ***show***:

gpart show,

позволяющую увидеть все разделы на всех подключенных дисках. А еще более подробная информация о разделах дисков выводится при помощи ***gpart list***. Чтобы уменьшить объем вывода, выводя информацию только об интересующем диске, следует указать в названных командах путь к нужному диску для ***show*** или имя диска для ***list.***

Задать новую таблицу разделов диска можно с помощью команды

gpart create –f x ***–s*** <тип_таблицы_разделов> <путь_к_диску>,

где в качестве типа таблицы разделов могут выступать 8 различных типов, в том числе, MBR и GPT.

Добавить новый раздел позволяет подкоманда ***add,*** в которой обязательным параметром является и тип файловой системы

gpart add –s 500m ***freebsd*** <путь_к_диску>,

например:

gpart add –t mbr /dev/ada2
(при правильно выполненной команде ответ системы будет >> ada2p2 added)

или

gpart add –t freebsd – s 500m /dev/ada2
(ответ системы - >> ada2s1 added).

После внесения изменений в таблицу разделов, например, после добавления нового раздела, следует выполнить команду

gpart commit <имя_диска>,

записывающую сделанные изменения на диск.

Инициализировать созданные разделы, выполнив команду ***newfs*** для каждого из добавленных разделов

newfs /dev/<имя_раздела>

например, ***newfs*** /dev/ada2s1.

Смонтировать (подключить в структуру директориев) созданные разделы для_таблиц разделов MBR или GPT, как в Linux, так и в Unix можно с помощью команды:

mount <подключаемый_раздел> <точка (директорий) подключения>,

например, ***mount*** /dev/ada2s1 /mnt/lhd3.

Для подключения флэшки следует смонтировать ее командой ***mount_msdosfs***, например, ***mount_msdosfs*** /dev/da0s1 /mnt/fla1.

Работа с дисками в системе Windows

При выполнении п. 9,а программы работы для создания разделов на диске в ОС семейств Windows следует воспользоваться утилитой DiskPart, вызываемой из командной строки.

(Следует учитывать, что возможности этой утилиты в Windows XP более ограничены по сравнению с последующими версиями Windows.)

Для определения имеющихся в системе дисков, необходимо выполнить команду ***list disk***. Для создания раздела на диске следует сначала выбрать требуемый диск с помощью команды ***select disk n***, где ***n*** – номер диска, на котором требуется создать раздел. После этого следует выполнить команду создания раздела, указав требуемый вид раздела: основной, расширенный или логический - и размер раздела в мегабайтах.

create partition [primary | extended] [size=].

Затем нужно установить фокус на созданный раздел (выбрать раздел) командой ***select partition n***, где ***n*** – номер раздела (предварительно можно просмотреть имеющиеся на диске разделы командой ***list partition***). Обычно, после создания нового раздела фокус на него устанавливается автоматически.

После чего с помощью команды ***format*** можно создать на выбранном разделе файловую систему:

format [***fs***=<тип системы>] [***label***=<”метка”>]

Назначить букву диску (фактически, смонтировать его) можно с помощью команды ***assign letter =*** <буква диска>. Имеется и разновидность данной команды: ***assign mount =*** <точка монтирования>. Перед таким назначением или монтированием следует выбрать требуемый том – отформатированный раздел – с помощью команды ***select volume***.

(Назначить букву, в принципе, можно и до форматирования раздела.)

Удалить ошибочно созданный раздел можно командой ***delete partition***, выбрав предварительно требуемый раздел командой select.

При выполнении пункта 9, б) создать каталог c:\FIO\ и сделать его виртуальным диском K следует использовать команду **subst**, вызываемую из командной строки (а не утилиты DiskPart)

<cmd> **subst** k: c:\fio.

Результатом выполнения этой команды будет назначение выбранной буквы заданной папке. Эту папку в ранних операционных системах Microsoft называли виртуальным диском.

Для систем, начиная с Windows7 (в Windows XP это не выполняется) поддерживается создание специальных типов файлов, называемых виртуальными дисками (.vhd, существуют и другие форматы, но они не поддерживаются в Windows 7). Создать такой диск можно командой ***create***, вызываемой в утилите DiskPart:

<diskpart> ***create vdisk file***=<"полное имя файла"> ***maximum***=<n> [***type***={fixed | expandable}] [***source***=<"полное имя файла">]

file=<"имя_файла"> – полное имя файла виртуального диска.
maximum=<n> – максимальный возможный объем виртуального диска в мегабайтах (МБ).

Необязательные параметры определяют формат диска (фиксированный или расширяемый) и первоначально копируемые на диск файлы другого виртуального диска.

После создания виртуального диска следует подключить его, выполнив команду ***attach***, выбрав, предварительно, этот диск командой ***select***, если после его создания выполнялись действия, которые сместили с него фокус:

select vdisk file=<"полное имя_файла">
attach vdisk [readonly] { [SD=<строка sddl>] | [usefilesd] }

Необязательные параметры определяют режим обращения к диску и дескрипторы безопасности (см. справку по команде ***create vdisk***).

Затем следует создать необходимые разделы (*create partition*) и файловые системы (*format*) на этом диске и присвоить им буквы и метки, как и на обычном диске

Выполняя пункт 9, в), проверить созданные файловые системы можно с помощью команд ***chkntfs***, если на диске используется файловая система NTFS, или ***chkdsk***, для любой файловой системы. Однако выполнение последней команды для дисков большого объема может занять значительное время. Можно получить дополнительную информациою с помощью утилиты ***fsutil***.

При выполнении пункта 9, г) установить соответствие пути в файловой системе заданному диску можно с помощью команды ***assign mount***=<”путь”>.

Работу с метками диска в пунктах 9, д) е) и ж) можно производить с помощью команды ***label***, вызываемой из командной строки, а также команды ***vol***.

Для преобразования томов с файловыми системами семейства FAT в тома с системой NTFS в п. 9, ж) используется вызываемая из командной строки команда ***convert***, в которой указывается преобразуемый диск, а затем по запросу подтверждается его метка.

Для работы с файлами и директориями при выполнении пунктов 9, и) и к) используются команды

***md* или *mkdir* (создание директория),**
***rd* или *rmdir* (удаление директория),**
***move* (перемещение директориев и файлов),**
***copy* (копирование файлов),**
***xcopy* (копирование файлов и директориев),**
***replace* (замещение файлов и директориев)**
***del* (удаление файлов),**
***erase* (удаление файлов),**
***edit* (редактор файлов),**
а также перенаправление вывода в файл: > или >> и далее – имя файла.

Для работы с потоками файлов NTFS из консоли можно использовать следующие команды:
- создание файла с потоком: type nul > somefile.txt:Stream
- запись в поток: echo "Something" >> somefile.txt:Stream
- чтение из потока: more < somefile:Stream
- копирование содержимого существующего файла в поток: type file1.txt >> somefile.txt:Stream
- копирование содержимого потока в файл: more < somefile.txt:Stream >> file2.txt.

5. Содержание отчета по практичесой работе

Отчет по работе должен содержать:
1. Описание использованных при выполнении работы команд
2. Примеры выполнения основных операций над дисками и файлами в операционных системах Windows и Linux.

6. Литература и ссылки на информационные ресурсы

- www.citforum.ru
- Man pages на русском (fsck, mkfs, fdisk, mount, umount, df)

Ссылка на руководство по командам на русском языке
http://citforum.ru/operating_systems/manpages/index.shtml

Сводка команд для работы с файлами и дисками

Для использования нового диска после его добавления необходимо:
- создать (заготовку) таблицу разделов
- создать раздел (ы) на диске
- зафиксировать изменения в таблице разделов диска
- создать на разделе файловую систему (отформатировать раздел) (для Unix это инициализация файловой системы, указанной при создании раздела)
- назначить букву разделу (тому) для Windows или смонтировать раздел для Unix и Linux

Разделы диска

	Действие	**Windows**	**Linux**	**Unix**
	Создать таблицу	**format**		**gpart create**
	Создать раздел	**create partition**	**fdisk n** (new) **cfdisk**	**gpart add** fdisk, bsdlabel
	Получить информацию о разделе	**list partition detail partition**	**parted print fdisk p** (print)	**gpart show** fdisk -s
	Уменьшить			
	Увеличить	**extend**		
	Удалить	**delete partition**	**fdisk d** (delete)	
	Записать изменения		**fdisk w** (write)	**commit**

Файловые системы (тома)

	Действие	**Windows**	**Linux**	**Unix**
	Создание/Инициализация	**format**	**mkfs.**<fs_type>	**newfs**
	Получить информацию о системе	**list volume detail volume**	**parted print**	
	Уменьшить	(**shrink**)		
	Увеличить	**extend**		
	Назначить букву	**assign**		
	Монтирование		**mount**	**mount**
	Размонтирование		**umount**	**umount**

Директории (папки)

	Действие	**Windows**	**Linux**	**Unix**
	Создать	**mkdir**, **md**	**mkdir**	**mkdir**
	Просмотреть	**dir**	**ls**	**ls**
	Переименовать	**rename**, **ren**	**mv**	
	Копировать	**xcopy**		
	Переместить		**mv**	
	Перейти в другой	**cd, chdir**	**cd**	**cd**
	Удалить	**rmdir, rd**	**rmdir**	**rmdir** *dir_name* rm -rf

Файлы

	Действие	**Windows**	**Linux**	**Unix**
	Создать		**touch**	
	Просмотреть	**type**	**cat, tail**	**cat, head, tail, more**
	Редактировать	**edit**	**nano, gedit**	**ee, vi, less (= vi)**, emac, vim
	Копировать	**copy**	**cp**	**cp** *src dst*
	Переместить	**move**		**mv** *src dst*
	Переименовать	**rename**, **ren**	**rename**	
	Удалить	**del, erase**	**rm**	**rm** *file_name*

Дополнительные

	Действие	**Windows**	**Linux**	**Unix**
	Задать права		**chmod**	**chmod**
	Помощь	**help**	**man, --help**	**man**
	Прервать команду	**Ctrl-C**	**Ctrl-C**	**Ctrl-C**
	Дата и время	**date, time**	**date**	**date**

Выход из редактора (vi), в который можно попасть по команде less <имя файла>, - по нажатию клавиш **Esc : q !** (без сохранения).

Для выхода из редактора **ee** нажать клавишу **Esc**, затем **Enter.** Если остались какие-либо не сохраненные данные, вам потребуется подтвердить выход, сохранив результат работы или оставив файл без изменения.

Практическая работа № 3

Управление процессами

1. Цель работы

Целью работы является получение практических навыков управления процессами и самостоятельной работы с документацией команд.

2. Общие сведения

Для управления процессами используются различные команды, позволяющие запускать процессы, приостанавливать и возобновлять их, устанавливать их приоритет, прекращать процессы.

Для задания типовых действий над процессами в ОС используются специальные команды, которые могут различаться в зависимости от операционной системы. Тем не менее, обычно часть команд являются общими для различных систем, основываясь на идеях межплатформенных взаимодействий систем и приложений.

2.1. Команды POSIX для работы с процессами *(должны быть во всех типах ОС, но их нет в Windows):*

at - запускает программы в определенное время
crontab – файл, содержащий таблицу расписаний запуска заданий
kill - прекращение выполнения процесса по PID процесса
nice - задает приоритет процесса перед его запуском
renice - изменяет приоритет работающего процесса
ps - выводит информацию о состоянии запущенных процессов
fg - перевод процесса из фонового режима
bg - продолжение выполнения фонового процесса, если он приостановлен нажатием <Ctrl+Z>

2.2. Команды LINUX для работы с процессами:

at - запускает программы в определенное время
atq - выводит список заданий, поставленных в очередь командой **at**
atrm - удаление задания из очереди команды **at**
/etc/crontab - таблицу расписаний запуска заданий - файл содержащий
kill - прекращение выполнения процесса по PID процесса
killall - прекращение выполнения процесса по имени процесса
nice - задает приоритет процесса перед его запуском
renice - изменяет приоритет работающего процесса
ps - выводит информацию о работающих процессах
top - выводит динамическую информацию о процессах
fg - вывод процесса из фонового режима
bg - продолжение выполнения фонового процесса, если он приостановлен нажатием <Ctrl+Z>
ipcs - взаимодействие процессов (разделяемая память, семафоры, сообщения)

Для получения более подробной информации, можно использовать **help** (например: **ps --help**), или документацию (например: **man ps**, для выхода - нажать **q**).

Запуск **фонового** процесса осуществляется так: **ps -x** &

При загрузке системы, автоматически загружаемые процессы, запускаются в фоновом режиме. Такие процессы называют "демонами". Они находятся в каталоге **/etc/rc.d/init.d/**.

Некоторые полезные комбинации клавиш:

<Ctrl+Z> - приостановить выполнение задания
<Ctrl+C> - завершить выполнение задания
<Ctrl+D> - завершить выполнение процедуры редактирования

Связывание процессов с помощью каналов

Запуск нескольких команд с передачей выходного потока следующей программе, "|" означает передачу выходного потока от первой программы ко второй.

ps -ax | more

запускается команда **ps -ax**, и передает выходной поток программе more которая запускается на выполнение.

Перенаправление ввода/вывода

Запуск команды с записью выходного потока в файл
ps -ax > test.txt
ps -ax > test.txt - добавит в конец файла

Способы задания группы команд:

command-1;command-2;command-3
{command-1;command-2} > test.txt

2.3. Команды Windows для работы с процессами:

Основную информацию о процессах можно получить, используя ***диспетчер задач.***

at - запуск программ в заданное время
Schtasks - настраивает выполнение команд по расписанию
Start - запускает определенную программу или команду в отдельном окне.
Taskkill - завершает процесс
Tasklist - выводит информацию о работающих процессах

Для получения более подробной информации, можно использовать центр справки и поддержки или команду **help** (например: **help at**)

command.com - запуск командной оболочки MS-DOS
cmd.exe - запуск командной оболочки Windows

3. Порядок выполнения практической работы

1. Запустить операционную систему семейства Linux/Unix.
2. Составить список команд работы с процессами, указав параметры их вызова.
3. Используя команды освоить особенности выполнения следующих видов операций с процессами: запуск, запуск по времени, изменение приоритетов, уничтожение процессов. Для этого:

а) создать задание на запуск процессов в определенное время, в определенную дату и с определенным приоритетом (в очереди на запуск), показать список заданий, удалить задание из списка;
б) рассмотреть порядок конфигурирования файла **/etc/crontab**;
в) выдать сигнал на прекращение процесса по ID и по имени;
г) запустить процесс с конкретным приоритетом
д) изменить приоритет конкретного процесса
е) вывести информацию о работающих процессах
ж) запустить процесс в фоновом режиме, выводить из него:
- выходной поток на передачу другой программе
- выходной поток процесса в файл.
з) вывести информацию о работающих процессах, с различным количеством столбцов и процессов
и) вывести динамическую информацию о процессах, сортировать эту информацию по столбцам, убирать и добавлять столбцы, менять приоритет процесса, уничтожать процесс.

4. Создать в виртуальной машине задание на запуск программы, указав минуту, час, день месяца, месяц, соответствующие запуску через количество минут, равное последней цифре шифра. Прописать задание в *crontab*.

5. Запустить ОС Windows
6. Вызвать диспетчер задач, проанализировать предоставляемую им в различных вкладках информацию.
7. Составить список команд работы с процессами, указав параметры их вызова.
8. Используя команды командной строки освоить особенности выполнения следующих видов операций с процессами: запуск, запуск по времени, изменение приоритетов, уничтожение процессов:
а) создать задание на запуск процессов в определенное время, показать список заданий, удалить задание из списка;
б) запустить определенную команду или программу в отдельном окне;
в) завершить процесс;
г) выводить информацию о работающих процессах и потоках;

9. Создать в виртуальной машине задание на запуск программы, указав минуту, час, день месяца, месяц, соответствующие запуску через количество минут, равное последней цифре шифра.

4. Указания к выполнению практической работы

Процессы Linux

При выполнении действий, указанных в пункте 3,а), следует использовать команду ***at***, которая позволяет поставить задание в очередь на исполнение в указанное время

```
at  время [дата] [+задержка]
```

например,

```
at  0815am Jan 16
at  8:15am Jan 16
at  now +1 day
at  5 pm Friday
```

С помощью этой же команды можно просматривать задания, вызвав ее с ключом –l (эквивалент команды ***atq***), и удалять задания, вызвав ее сключем –r и уменем удаляемого задания (эквивалент команды ***atrm***).

Для остановки запуска ОС следует создать небольшую программу, содержащую ожидание нажатия клавиши и поместить ее в директорий /etc/init.d, в котором размещаются скрипты запуска служб.

Для того, чтобы эта программа могла вывести в консольном режиме сообщение во время запуска ОС, целесообразно включить ее в начало сценария запуска графической оболочки /etc/init.d/gdm3, добавив в его начало две строки

```
stty -cread -F /dev/tty1
/etc/init.d/<имя_программы>
```

Процессы Windows

При выполнении действий, указанных в пункте 8,а), следует использовать команду ***at***, которая позволяет поставить задание в очередь на исполнение в указанное время

5. Отчет по практической работе

Отчет по работе должен содержать:

1. Описание использованных при выполнении работы команд
2. Примеры выполнения основных операций над процессами в операционных системах Windows и Linux.

6. Литература и ссылки на информационные ресурсы

1. Современные операционные системы. 3-е изд., Э. Танненбаум, 2010 год, 1120 стр.
2. Операционные системы. Разработка и реализация, Э. Таненбаум, А. Вудхалл., 2007, 3-е изд., СПб.: Питер, 704 стр.
3. Сетевые операционные системы: Учебник для вузов, 2-е изд. Н. А. Олифер, В. Г. Олифер, 2009 год, 672 стр.

- www.citforum.ru
- Man pages на русском (at, cron, crontab, kill, killall, nice, renice, nohup, ps)
- Выполнение, останов и повторный запуск процессов (batch, at, ps, kill, fg, bg)
- Управление процессами (kill, nice, top, nohup)
- Управление процессами (at, cron, crontab, ps, kill, proc, init)
- Исследуем процессы. Часть 1 (w, ps)
- Исследуем процессы. Часть 2 (kill)
- Избранные команды Unix (kill, nice, ps, who)

Ссылка на руководство по командам на русском языке
http://citforum.ru/operating_systems/manpages/index.shtml

atd – запускает задания, поставленные в очередь на последующее исполнение
batch – выполняет команды, когда это позволяет загрузка системы (средняя загрузка менее 1,5 или величины, указанной при вызове **atd**)

```
at, batch, atq, atrm - queue, examine or delete jobs for later execution
SYNOPSIS
       at [-V] [-q queue] [-f file] [-mMlv] timespec...
       at [-V] [-q queue] [-f file] [-mMkv] [-t time]
       at -c job [job...]
       atq [-V] [-q queue]
       at [-rd] job [job...]
       atrm [-V] job [job...]
       batch
       at -b
```

Отредактировать пользовательский файл можно, вызвав редактор командой crontab -e. При этом в качестве редактора будет вызван указанный в переменной окружения VISUAL или EDITOR. По завершении редактирования файл будет помещен в директорий /etc/spool/cron/crontabs/<имя_пользователя>.

(Типовыми редакторами могут быть /bin/ed, /bin/nano, /usr/bin/vim.tiny)

Формат записи в системных и пользовательских файлах группы cron следующий:

m h dom mon dow user command

(# мин час число мес день_недели пользователь команда)

17 * * * * root cd / && run-parts --report /etc/cron.hourly

25 6 * * * root test -x /usr/sbin/anacron || (cd / && run-parts --report /etc/cron.daily)

47 6 * * 7 root test -x /usr/sbin/anacron || (cd / && run-parts --report /etc/cron.weekly)

52 6 1 * * root test -x /usr/sbin/anacron || (cd / && run-parts --report /etc/cron.monthly)

Редактирование общесистемного файла /etc/crontab осуществляется с помощью обычного редактора.

Практическая работа № 4

Объекты синхронизации Windows

1. Цель работы

Целью практической работы является получение практических навыков синхронизации процессов многопоточных приложений в Windows.

2. Общие сведения

Объекты синхронизации – это такие объекты ОС, которые создаются и управляются программно, являются общими для всех потоков в системе (некоторые - для потоков, принадлежащих одному процессу) и используются для координирования доступа к ресурсам. В качестве ресурсов может выступать все, что может быть общим для двух и более потоков - файл на диске, порт, запись в базе данных, объект GDI, и даже глобальная переменная программы (которая может быть доступна из потоков, принадлежащих одному процессу).

Основными объектами синхронизации являются:
- взаимоисключение (mutex),
- критическая секция (critical section),
- событие (event),
- семафор (semaphore).

Каждый из них реализует свой механизм синхронизации, который используется в конкретных случаях. Объектами синхронизации могут выступать и сами процессы и потоки (если они честно ожидают завершения другого потока или процесса), файлы, коммуникационные устройства, консольный ввод и уведомления об изменении.

Каждый объект синхронизации может находиться в так называемом сигнальном состоянии. Для каждого типа объектов это состояние имеет различный смысл. Потоки могут проверять текущее состояние объекта и/или ждать изменения этого состояния и таким образом согласовывать свои действия. Очень важно, что когда поток работает с объектами синхронизации (создает их, изменяет состояние) система гарантированно не прервет его выполнения, пока он не завершит это действие. Таким образом, все конечные операции с объектами синхронизации являются атомарными (неделимыми).

Реальной связи объектов синхронизации с ресурсами, конечно, нет. Они не предотвращают доступ к ресурсу, а только сообщают о возможности работы с ресурсом или необходимости ожидания доступа к нему.

(Help Windows:) Критические секции обеспечивают синхронизацию, аналогичную той, которую дают и мьютексы, за исключением того, что критическая секция может использоваться только потоками одного процесса. События, мьютексы и семафоры также могут использоваться в однопроцессном приложении, но критическая секция работает несколько быстрее и эффективнее при взаимоисключающей синхронизации. Подобно мьютексу, критической секцией в каждый момент времени может владеть только один поток, что делает ее полезной для защиты разделяемого ресурса от одновременного доступа. Например, процесс может использовать критическую секцию, чтобы избежать одновременной модификации глобальной структуры данных более, чем одним потоком.

Работа с объектами синхронизации

Чтобы создать тот или иной объект синхронизации, производится вызов специальной функции WinAPI типа Create... (напр. CreateMutex). Этот вызов возвращает дескриптор объекта (HANDLE), который может использоваться всеми потоками, принадлежащими данному процессу. Есть возможность получить доступ к объекту синхронизации из другого процесса - либо унаследовав дескриптор этого объекта, либо, что предпочтительнее, воспользовавшись вызовом функции открытия объекта (Open...). После этого вызова процесс получит дескриптор, который в дальнейшем можно использовать для работы с объектом. Объекту, если только он не предназначен для использования внутри одного процесса, обязательно присваивается имя. Имена всех объектов должны быть различны (даже если они разного типа). Нельзя, например, создать событие и семафор с одним и тем же именем.

По имеющемуся дескриптору объекта можно определить его текущее состояние. Это делается с помощью т.н. ожидающих функций. Чаще всего используется функция WaitForSingleObject. Эта функция принимает два параметра, первый из которых - дескриптор объекта, второй - время ожидания в мсек. Функция возвращает WAIT_OBJECT_0, если объект находится в

сигнальном состоянии, WAIT_TIMEOUT - если истекло время ожидания, и WAIT_ABANDONED, если объект-взаимоисключение не был освобожден до того, как владеющий им поток завершился. Если время ожидания указано равным нулю, функция возвращает результат немедленно, в противном случае она ждет в течение указанного промежутка времени. В случае, если состояние объекта станет сигнальным до истечения этого времени, функция вернет WAIT_OBJECT_0, в противном случае функция вернет WAIT_TIMEOUT.
Если в качестве времени указана символическая константа INFINITE, то функция будет ждать неограниченно долго, пока состояние объекта не станет сигнальным.
Если необходимо узнавать о состоянии сразу нескольких объектов, следует воспользоваться функцией WaitForMultipleObjects.
Чтобы закончить работу с объектом и освободить дескриптор вызывается функция CloseHandle.
Очень важен тот факт, что обращение к ожидающей функции блокирует текущий поток, т.е. пока поток находится в состоянии ожидания, ему не выделяется процессорного времени.

Взаимоисключения

Объекты-взаимоисключения (мьютексы, mutex - от MUTual EXclusion) позволяют координировать взаимное исключение доступа к разделяемому ресурсу. Сигнальное состояние объекта (т.е. состояние "установлен") соответствует моменту времени, когда объект не принадлежит ни одному потоку и его можно "захватить". И наоборот, состояние "сброшен" (не сигнальное) соответствует моменту, когда какой-либо поток уже владеет этим объектом. Доступ к объекту разрешается, когда поток, владеющий объектом, освободит его.

Для того, чтобы объявить взаимоисключение принадлежащим текущему потоку, надо вызвать одну из ожидающих функций. Поток, которому принадлежит объект, может его "захватывать" повторно сколько угодно раз (это не приведет к самоблокировке), но столько же раз он должен будет его освобождать с помощью функции ReleaseMutex.

События

Объекты-события используются для уведомления ожидающих потоков о наступлении какого-либо события. Различают два вида событий - с ручным и автоматическим сбросом. Ручной сброс осуществляется функцией ResetEvent. События с ручным сбросом используются для уведомления сразу нескольких потоков. При использовании события с автосбросом уведомление получит и продолжит свое выполнение только один ожидающий поток, остальные будут ожидать дальше.

Функция CreateEvent создает объект-событие, SetEvent - устанавливает событие в сигнальное состояние, ResetEvent- сбрасывает событие. Функция PulseEvent устанавливает событие, а после возобновления ожидающих это событие потоков (всех при ручном сбросе и только одного при автоматическом), сбрасывает его. Если ожидающих потоков нет, PulseEvent просто сбрасывает событие.

Семафоры

Объект-семафор - это фактически объект-взаимоисключение со счетчиком. Данный объект позволяет "захватить" себя определенному количеству потоков. После этого "захват" будет невозможен, пока один из ранее "захвативших" семафор потоков не освободит его. Семафоры применяются для ограничения количества потоков, одновременно работающих с ресурсом. Объекту при инициализации передается максимальное число потоков, после каждого "захвата" счетчик семафора уменьшается. Сигнальному состоянию соответствует значение счетчика больше нуля. Когда счетчик равен нулю, семафор считается не установленным (сброшенным).

Критические секции

Объект-критическая секция помогает программисту выделить участок кода, где поток получает доступ к разделяемому ресурсу, и предотвратить одновременное использование ресурса. Перед использованием ресурса поток входит в критическую секцию

(вызывает функцию EnterCriticalSection). Если после этого какой-либо другой поток попытается войти в ту же самую критическую секцию, его выполнение приостановится, пока первый поток не покинет секцию с помощью вызова LeaveCriticalSection. Похоже на взаимоисключение, но используется только для потоков одного процесса.

Существует также функция TryEnterCriticalSection, которая проверяет, занята ли критическая секция в данный момент. С ее помощью поток в процессе ожидания доступа к ресурсу может не блокироваться, а выполнять какие-то полезные действия.

Защищенный доступ к переменным

Существует ряд функций, позволяющих работать с глобальными переменными из всех потоков не заботясь о синхронизации, т.к. эти функции сами за ней следят. Это функции InterlockedIncrement/InterlockedDecrement, InterlockedExchange, InterlockedExchangeAdd и InterlockedCompareExchange. Например, функция InterlockedIncrement увеличивает значение 32-битной переменной на единицу - удобно использовать для различных счетчиков. Более подробно об этих функциях см. в документации.

Синхронизация в MFC

Библиотека MFC содержит специальные классы для синхронизации потоков (CMutex, CEvent, CCriticalSection и CSemaphore). Эти классы соответствуют объектам синхронизации WinAPI и являются производными от класса CSyncObject. Чтобы понять, как их использовать, достаточно просто взглянуть на конструкторы и методы этих классов - Lock и Unlock. Фактически эти классы - всего лишь обертки для объектов синхронизации.

Есть еще один способ использования этих классов - написание так называемых потоково-безопасных классов (thread-safe classes). Потоково-безопасный класс - это класс, представляющий какой либо ресурс в вашей программе. Вся работа с ресурсом осуществляется только через этот класс, который содержит все необходимые для этого методы. Причем класс спроектирован таким образом, что его методы сами заботятся о синхронизации,

так что в приложении он используется как обычный класс. Объект синхронизации MFC добавляется в этот класс в качестве закрытого члена класса, и все функции этого класса, осуществляющие доступ к ресурсу, согласуют с ним свою работу.

С классами синхронизации MFC можно работать как напрямую, используя методы Lock и Unlock, так и через промежуточные классы CSingleLock и CMultiLock (хотя на мой взгляд, работать через промежуточные классы несколько неудобно. Но использование класса CMultiLock необходимо, если вы хотите следить за состоянием сразу нескольких объектов).

Не используя возможности многозадачности, которые предоставляет Windows, программы теряют преимущества этой ОС.

Требования:

1. Использование только вызовов API
2. Желательно описание внешних вызовов ОС производить непосредственно в своем коде (разрешено пользоваться описанием windows.pas)
3. Обработка ошибок и их анализ по каждому вызову функций API
4. Остановка исследуемого процесса по вводу символа «q» с клавиатуры

Конечный результат:

1. Получить требуемую в варианте величину.
2. Пакетный файл компиляции.
3. Отчет с выводами и анализом результатов. В анализ должны входить оценки вероятностных показателей измеренной в варианте величины.
4. Файл с трассой работы объекта синхронизации, представленной в удобном для чтения формате.
5. Исходный код программы на дискете.

Варианты:

1. Реализовать схему «оповещения» трех ожидающих потоков о помещении в очередь (произвольного формата) некоторого сообщения на основе объекта синхронизации «событие». Генерируют события в очередь два потока, которые имеют приоритет на уровень ниже, чем ожидающие. Посчитать среднее число обработанных каждым потоком сообщений. Очередь должна быть защищена критической секцией.
2. Реализовать критическую секцию на основе объекта синхронизации типа «событие». Продемонстрировать функционирование на примере защиты реализованной критической секцией некоторой очереди сообщений произвольного формата.
3. Отобразить файл размера 32К в память. Реализовать транспортировку данных из отображения в файл на диске (один поток). Два других потока записывают некоторые данные в это отображение. Синхронизация доступа к данным отображения осуществляется при помощи объекта синхронизации типа «мьютекс».
4. Реализовать семафор на основе объекта синхронизации «событие». Количественные характеристики семафора должны быть аналогичны соответствующему объекту ядра типа семафор.
5. Реализовать моделирование «тупика» на примере работы с критическими секциями. Временные параметры модели определяются до начала моделирования (произвольны, но обязательно изменяемы). Провести анализ вероятности тупика, если она равна 1, то обосновать.
6. Реализовать моделирование «гонок» на примере работы с критическими секциями. Временные параметры модели определяются до начала моделирования (произвольны, но обязательно изменяемы). Провести анализ вероятности тупика, если она равна 1, то обосновать.
7. (со звездочкой) Реализовать моделирование «столкновений» на примере работы с критическими секциями. Временные параметры модели определяются до начала моделирования (произвольны, но обязательно изменяемы). Провести анализ вероятности тупика, если она равна 1, то обосновать.

8. Реализовать мьютекс на основе объекта синхронизации «событие». Количественные характеристики мьютекс должны быть аналогичны соответствующему объекту ядра типа мьютекс.
9. Организовать совместное использование мьютекса и события для защиты добавления элемента в очереди. Очередь произвольного формата. Событие сигнализирует чтение из очереди, а мьютекс - запись. Требуется снять статистику по времени использования объектов ядра.
10. Реализовать на мьютексе семафор. Количественные характеристики семафора должны быть аналогичны соответствующему объекту ядра типа семафор.

Практическая работа № 5

Командные файлы

1. Цель работы

Изучение порядка написания и получение практических навыков создания и использования командных файлов.

2. Основные теоретические положения

Командный (пакетный, бат-) файл представляет собой текстовый файл с набором инструкций командному процессору cmd.exe, позволяющий автоматически выполнить записанный в текстовом файле набор команд ОС.

В качестве инструкций могут выступать как имена исполняемых файлов, так и специальные инструкции командного процессора (операционной системы).

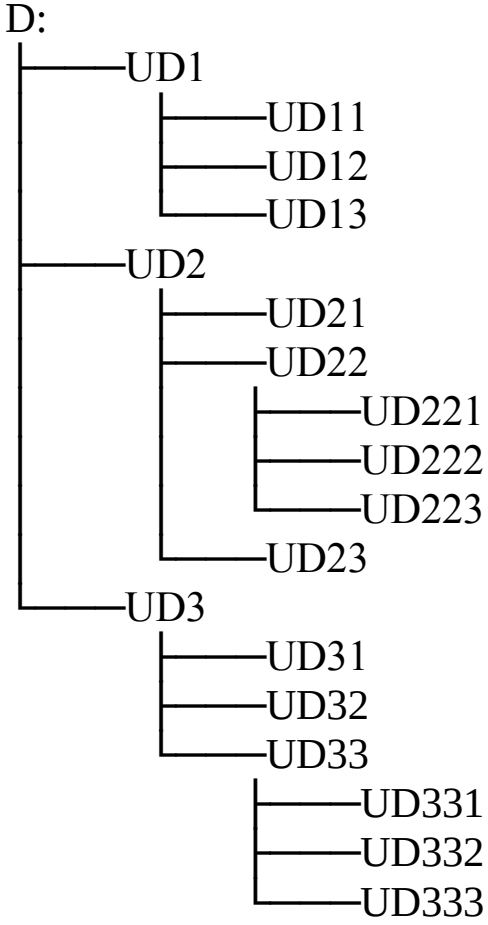
Командная оболочка ОС Windows использует для перевода введенной команды в формат, понятный компьютеру, и выполнения команды интерпретатор команд cmd.exe.

3. Порядок выполнения работы

3.1. Работа с каталогами

1.1. Пользуясь командами cmd.exe для работы с каталогами, создать заданное дерево каталогов.

Перед выполнением задания необходимо перейти на диск D:, создать каталог UD, перейти в этот каталог, создать директорий UD_FIO (где FIO инициалы студента, записанные латинскими буквами), перейти в этот каталог и уже в нем выполнять задание практической работы.



1.2. Написать командный файл cmf1.bat, выполнение которого создаст заданное дерево каталогов и выведет на экран содержание каталога UD_FIO, включая все подкаталоги.

1.3. Запустить командный файл на исполнение и проверить его работу.

1.4. Написать командный файл com2.bat, выполнение которого создаст в директории UD_FIO файлы, перечисленные в ниже:

f1.txt, f2.txt, f3.txt,
f4.pas, f5.pas, f6.pas,
f7.cpp, f8.cpp, f9.cpp,
f10.bat, f11.bat, 12.bat,
f13.exe, f14.exe, f15.exe,
f16.gif, f17.gif, f18.gif,
f19.com, f20.com, f21.com,

f22.tmp, f23.tmp, f24.tmp.

В качестве содержимого файлов ввести текст: FILE 1 для f1.txt, FILE 11 для f11.bat и т. д.

1.5. В командном файле предусмотреть выполнение следующих операций:

- вывести на экран содержимое каталога UD_FIO, включая все подкаталоги.

- вывести на экран все временные файлы из директории UD_FIO. Показать дату и время создания файлов, а также их размер.

- вывести на экран все программные файлы из директории UD_FIO, показать только имена файлов.

- вывести на экран все программные файлы из директории UD_FIO, показать только имена файлов.

Из каталога D:\UD_FIO скопировать:

- в каталог D:\UD_FIO\UD2\UD22\UD222 все текстовые файлы;

- в каталог D:\UD_FIO\UD2\UD22\UD223 файлы рисунков;

- в каталог D:\UD_FIO\UD2\UD22\UD221 все командные файлы.

Из каталога D:\UD_FIO скопировать одной командой FOR все программные файлы (.cpp и .pas) в каталог D:\UD_FIO\UD3\UD32.

Из каталога UD_FIO переместить:

- в каталог D:\UD_FIO\UD1\UD12\ исходные модули на языке C++;

- в каталог D:\UD_FIO\UD1\UD13\ исполняемые машинные программы.

Удалить временные файлы из каталога UD_FIO.

Вывести на экран содержимое всего дерева каталогов, начиная с директории UD_FIO, показав только имена файлов.

1.6. Запустить командный файл на исполнение и проверить его работу.

3.2. Работа с файлами

2.1. Пользуясь внутренними командами (copy, echo) cmd.exe, создать в директории UD_FIO файлы, перечисленные в пункте 1.4 порядка выполнения работы.

2.2. Пользуясь командами echo, more, к файлам f1.txt, f2.txt присоединить по два потока: к файлу f1.txt – поток 1 с содержанием: potok1, поток 2 с содержанием: potok2; к файлу f2.txt – поток 3 с содержанием: potok3, поток 4 с содержанием: potok4.
2.3. Распечатать на экране содержимое неименованных потоков для f1.txt и f2.txt.
2.4. Распечатать на экране содержимое именованных потоков для f1.txt и f2.txt.
2.5. Скопировать f1.txt и f2.txt на мобильный носитель (дискету, карту флэш-памяти).
Распечатать на экране содержимое именованных потоков для f1.txt и f2.txt, находящихся на мобильном носителе.
2.6. Вывести на экран содержание всего дерева каталогов, начиная с директории UD_FIO, показать только имена файлов. Привести вид экрана.
2.7. Все действия оператора и результаты работы введенных команд документировать.

3.3. Резервное копирование материалов

3.1. Пользуясь текстовым редактором, создать командный файл com3.bat, выполнение которого реализует резервное копирование в каталог backuplab2 всех файлов и подкаталогов учебного каталога UD_FIO.
3.2. Вывести на экран содержание всего каталога backuplab1.
3.3. Все действия оператора и результаты работы введенных команд документировать с помощью операции «пометить» (в окне командной строки) и операции «вставить» (в окне текстового редактора).

3.4. Присвоение значений внутренним переменным. Параметры командных файлов

4.1. Пользуясь внутренними командами cmd.exe для работы с каталогами (C: − переход на диск C, dir (путь) (имя_файла) (/p) (/w), cd путь, md путь, rd путь), перейти на диск D:, создать каталог U:\ UD_FIO.

4.2. Пользуясь текстовым редактором (Блокнот, Word), создать командный файл com31.bat, выполнение которого реализует следующую последовательность операций:

a. Присвоить внутренней переменной p1 численное значение 1-го параметра командного файла;
b. Присвоить внутренней переменной p2 численное значение 2-го параметра командного файла;
c. Присвоить внутренней переменной p3 значение суммы p1+p2;
d. Вывести на экран значения переменных p1, p2, p3 и сообщение об имени отработавшего бат-файла;
e. Направить результаты работы bat-файла в файл протокола работы командного файла;
f. Имя файла протокола сформировать как конкатенацию: (имя bat-файла)_log1.txt.

4.3. Запустить командный файл на исполнение и проверить его работу.
4.4. Используя команду SET, проконтролировать появление переменных p1, p2 и p3 в списке переменных окружения ОС.
4.5. Исследовать работу bat-файла при неопределенных параметрах.
4.6. Используя команду SET, исследовать поведение переменных p1, p2 и p3 в списке переменных окружения ОС.
4.7. Все действия оператора и результаты работы введенных команд документировать.
4.8. Содержимое файла протокола работы bat-файла привести в отчете.
4.9. Сведения о командах можно получить из справочной системы ОС, используя запросы следующего вида: set /?

3.5. Использование операторов if и goto для исключения возможности работы командного файла с неопределенными параметрами

5.1. Пользуясь внутренними командами cmd.exe, создать командный файл kom32.bat как копию kom31.bat.
5.2. Пользуясь текстовым редактором, отредактировать командный файл kom32.bat таким образом, чтобы его запуск с неопределенными параметрами не приводил к изменению значений внутренних переменных p1, p2 и p3. В этом случае запуск

командного файла должен выводить на экран (и в файл протокола исполнения) сообщение parameters absent.
5.3. Используя команду SET, проконтролировать поведение переменных p1, p2 и p3 в списке переменных окружения ОС.

4. Указания к выполнению работы

Сведения о поддерживаемых в командной строке командах можно получить, используя запрос help из командной строки ОС, сведения о порядке использования любой из команд можно получить из справочной системы ОС, используя запрос следующего вида md /? (в данном случае – все о команде md).

Все действия оператора и результаты работы введенных команд документировать с помощью операции «пометить» (в окне командной строки) и операции «вставить» (в окне текстового редактора).

Для создания файлов и задания их содержимого следует использовать комбинацию команду **echo** и перенаправление вывода этой команды в требуемый файл >

5. Содержание отчета

1. Задание на практическую работу.
2. Распечатки командных файлов и результатов их выполнения по каждому пункту задания.
3. Выводы по практической работе.
4. Описание используемых в работе команд.
5. Распечатки файлов протоколов исполнения командных файлов.

СОДЕРЖАНИЕ

Введение.....3
Практическая работа № 1 4
Практическая работа № 2 9
Практическая работа № 3 25
Практическая работа № 4 31
Практическая работа № 5 39