Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный технологический университет»

Факультет информационных технологий

Кафедра информационных систем и технологий

**Операционные системы и системное программирование**

Реферат

Выполнил:

студент 3 курса 4 группы

специальности ПОИТ

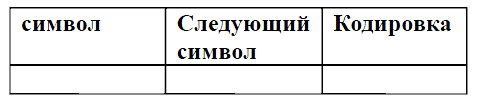
Матюх А.А.

Минск 2020

1. **Методы сжатия информации**

В основе идеи по сжатию информации лежат алгоритмы по замене часто встречающиеся комбинации более короткими (алгоритм Хаффмана), либо замене более длинные последовательности более короткими (метод Лемпеля-Зива). Это касается в основном текстовой информации, в изображениях же сжатие основано на алгоритмах частичной потери информации, которые минимально влияют на качество изображения.

В системах Windows в основном используется метод Лемпеля-Зива и его различные модификации. Он основан на таблице типа:



Где разбирая сообщение, мы помещаем символы в столбец “символ”. Если же при разборе оказывается, что текущий символ уже есть в столбце “символ”, тогда объединяем его со следующими символами до тех пор, пока значение не станет уникальным. Далее в столбец “кодировка” нумеруются строки. Нумерацию переводим в двоичный вид.

Завершив заполнение таблицы, соотносим символы сообщения со строкой и столбцом “символ”, постепенно заменяя на значения из “кодировки”.

Для работы же с компрессированными файлами по Лемпелю-Зиву необходима библиотека “LZ32”:

#pragma comment(lib, "LZ32")

Для чтения файла, сжатого по Лемпелю-Зиву, создается буфер в памяти, используемый как рабочая область при выполнении алгоритма:

(LPOFSTRUCT)malloc(sizeof(LPOFSTRUCT));

Открытие файла:

hdest = LZOpenFile(Destination, OpenStruct, OF\_CREATE);

Декомпрессия:

LZCopy(hsource, hdest);

1. **Сетевые Технологии**

В настоящий момент в сетевых технологиях используется архитектура клиент-сервер, которая подразумевает разделение ролей.

Сервер – компьютер сети, предоставляющий свои программные и аппаратные ресурсы пользователям сети для хранения данных, выполнения программ и других услуг (например, доступ к общей базе данных, совместное использование устройств ввода/вывода, организацию взаимодействия пользователей и др.).

Клиент – компонент архитектуры "клиент – сервер", пользующийся услугами сервера. Часто в качестве клиента выступают программы, имеющие доступ к информационным ресурсам или устройствам сервера. Для подключения к серверу пользователь рабочей станции должен получить собственное регистрационное имя и пароль.

За счет этого в разработке программного обеспечения и обработке информации появилось деление на Back-end и Front-end.

Для клиента не важна операционная система, клиент по своей сути кроссплатформенен. Для серверов же разрабатываются специальные версии ОС, например: Windows NT, OS/2, Novell NetWare, UNIX и Mac OS с сетевыми службами AppleShare и AppleTalk.

Сетевой операционной системе важны 3 параметра: многопоточность, многозадачность и многопроцессорность.

Многопроцессорность может быть симметричной и асимметричной.

При симметричной обработке любой процесс может быть поручен любому, в данный момент свободному процессору.

При асимметричной обработке нагрузка распределяется так, что один или несколько процессоров обслуживают только операционную систему, а остальные работают только с приложениями.

Многопоточность основана на том, что процессор работает с невероятной скоростью, измеряемой тактами. Эти такты выполняются независимо от того, занят процессор или нет.

При многопоточной обработке процесс подразделяется на потоки, каждый из которых выполняется микропроцессором по отдельности. Планирование потоков осуществляется операционной системой.

Многозадачность основана на том, что время процессора предоставляется каждому процессу отдельно, а человеку кажется, что эти процессы идут параллельно.

1. **Сервисы**

Сервисы – фоновые процессы в оперативной памяти. Их количество достаточно велико, так что они реально могут тормозить работу компьютера. То есть это любое программное обеспечение, которое устанавливается в операционную систему как фоновый процесс. Для установки сервиса можно воспользоваться утилитой sc и командой в ней:

sc create [service name] binpath= [exe path]

Для удаления в этой же утилите есть команда:

sc delete [service name]

Рассмотрим пример написания сервиса:

SERVICE\_TABLE\_ENTRY ServiceTable[1]; // сущность сервисной таблицы

ServiceTable[0].lpServiceName //Имя сервиса

ServiceTable[0].lpServiceProc //метод исполняемый сервисом

Функция инициализации сервиса:

DWORD ServiceInitialization(DWORD argc, LPTSTR \*argv, DWORD \*specificError)

Метод, исполняемый сервисом, должен относиться к классу WINAPI, пример:

void WINAPI ServiceStart(DWORD argc, LPTSTR \*argv)

Запуск сервиса:

SERVICE\_STATUS serviceStatus.dwCurrentState = SERVICE\_START\_PENDING;

Остановка сервиса:

serviceStatus.dwCurrentState = SERVICE\_STOPPED;

Сервис регистрируется при запуске StartServiceCtrlDispatcher. Основная процедура сервиса это ServiceStart.

Логика этой функции проста. Сначала регистрируем функцию которая будет обрабатывать управляющие запросы от SCM (диспетчера сервисов), например, запрос на остановку. Регистрация производиться при помощи функции RegisterServiceCtrlHandler.

1. **Рисование на форме. Графика**

В приложениях Windows Win32 Project есть два способа отображения графики: рисование и изображения.

Для рисования есть обработчик событий WM\_PAINT:

hdc = BeginPaint(hWnd, &ps); //начать отрисовку

GamePaint(hdc, hWnd); //метод выполняющий рисование

EndPaint(hWnd, &ps); //окончание отрисовки

Само же рисование производится за счет отрисовки простых фигур и работы с точками:

TextOut(hDC, 10, 10, TEXT("Oleg German"), 11); //отрисовка текста

MoveToEx(hDC, 10, 25, NULL); //смена текущей позиции точки

LineTo(hDC, 100, 25); //отрисовка линии начиная с текущей точки

RECT rect;

GetClientRect(hWnd, &rect);

Rectangle(hDC, 34, 50, 54, 70); //отрисовка прямоугольника по точкам

Ellipse(hDC, 300, 200, 580, 380); //отрисовка эллипса по точкам

DrawText(hDC, TEXT("OLEG GERMAN"), -1, &rect, DT\_SINGLELINE |DT\_CENTER|DT\_VCENTER); //отрисовка текста в прямоугольнике

Для отображения изображения используется обработчик событий WM\_PAINT:

BeginPaint(hWnd, &ps); //начать отрисовку

CreateCompatibleDC(hdc); //создание контекста рисования

SelectObject(hdcMem, hbitmap); //получение объекта контекста

GetObject(hbitmap, sizeof(bitmap), &bitmap); //получения битовой карты

BitBlt(hdc, 0, 0, bitmap.bmWidth, bitmap.bmHeight, hdcMem, 0, 0, SRCCOPY); //Отрисовка битовой карты

DeleteDC(hdcMem);//Удаление / освобождение контекста

EndPaint(hWnd, &ps); //окончание отрисовки

Для получения изображение используется битовая карта, то есть просто набор бит подразумевающий под собой изображение из фала:

hbitmap = (HBITMAP)LoadImage(hInst, L[image path], IMAGE\_BITMAP, 0, 0, LR\_LOADFROMFILE);