УВАЖАЕМЫЕ СТУДЕНТЫ! Законспектируйте в своей рабочей тетради по дисциплине приведенную лекцию (объемом 4-5 страницы), ответьте *письменно* на контрольные вопросы.

Результаты работы, фотоотчет, предоставить преподавателю на e-mail: r.bigangel@gmail.com до 06.02.2023.

При возникновении вопросов по приведенному материалу обращаться по следующему номеру телефона: (072)111-37-59, (Viber, WhatsApp), vk.com: https://vk.com/daykini

ВНИМАНИЕ!!! При отправке работы, не забывайте указывать ФИО студента, наименование дисциплины, дата проведения занятия (по расписанию).

Лекция №5.

Тема: Принципы построения интерфейсов ОС. Виды интерфейсов

Напомним, что ОС всегда выступает как интерфейс между аппаратурой компьютера и пользователем с его задачами. Под интерфейсами операционных систем здесь и далее следует понимать специальные интерфейсы системного и прикладного программирования, предназначенные для выполнения следующих задач:

Управление процессами, которое включает в себя следующий набор основных функций:

- запуск, приостанов и снятие задачи с выполнения;
- задание или изменение приоритета задачи;
- взаимодействие задач между собой (механизмы сигналов, семафорные примитивы, очереди, конвейеры, почтовые ящики);
- RPC (remote procedure call) — удаленный вызов подпрограмм.

Управление памятью:

- запрос на выделение блока памяти;
- освобождение памяти;
- изменение параметров блока памяти (например, память может быть заблокирована процессом либо предоставлена в общий доступ);
- отображение файлов на память (имеется не во всех системах).

Управление вводом/выводом:

- запрос на управление виртуальными устройствами (напомним, что управление вводом/выводом является привилегированной функцией самой ОС, и никакая из пользовательских задач не должна иметь возможности непосредственно управлять устройствами);
- файловые операции (запросы к системе управления файлами на создание, изменение и удаление данных, организованных в файлы).

Здесь мы перечислили основные наборы функций, которые выполняются ОС по соответствующим запросам от задач. Что касается пользовательского интерфейса операционной системы, то он реализуется с помощью специальных программных модулей, которые принимают его команды на соответствующем языке (возможно, с использованием графического интерфейса) и транслируют их в обычные вызовы в соответствии с основным интерфейсом системы. Обычно эти модули называют интерпретатором команд. Так, например, функции такого интерпретатора в MS-DOS выполняет модуль COMMAND.COM. Получив от пользователя команду, такой модуль после лексического и синтаксического анализа либо сам выполняет действие, либо, что случается чаще, обращается к другим модулям ОС, используя механизм API. Надо заметить, что в последние годы большую популярность получили графические интерфейсы (GUI), в которых задействованы соответствующие манипуляторы типа «мышь» или «трекбол». Указание курсором на объекты и щелчок (клик) или двойной щелчок по соответствующим клавишам приводит к каким-либо действиям — запуску программы, ассоциированной с указываемым объектом, выбору и/или активизации пунктов меню и т. д. Можно сказать, что такая интерфейсная подсистема транслирует «команды» пользователя в обращения к ОС.

Поясним также, что управление GUI — частный случай задачи управления вводом/выводом, не являющийся частью ядра операционной системы, хотя в ряде случаев разработчики ОС относят функции GUI к основному системному API.

Следует отметить, что имеются два основных подхода к управлению задачами. Так, в одних системах порождаемая задача наследует все ресурсы задачи-родителя, тогда как в других системах существуют равноправные

отношения, и при порождении нового процесса ресурсы для него запрашиваются у операционной системы.

Обращения к операционной системе, в соответствии с имеющимся API, может осуществляться как посредством вызова подпрограммы с передачей ей необходимых параметров, так и через механизм программных прерываний. Выбор метода реализации вызовов функций API должен определяться архитектурой платформы.

Так, например, в операционной системе MS-DOS, которая разрабатывалась для однозадачного режима (поскольку процессор 18086 не поддерживал мультипрограммирование), использовался механизм программных прерываний. При этом основной набор функций API был доступен через точку входа обработчика int 21h.

В более сложных системах имеется не одна точка входа, а множество — по количеству функций API. Так, в большинстве операционных систем используется метод вызова подпрограмм. В этом случае вызов сначала передается в модуль API (например, это библиотека времени выполнения), который и перенаправляет вызов соответствующим обработчикам программных прерываний, входящим в состав операционной системы. Использование механизма прерываний вызвано, главным образом, тем, что при этом процессор переводится в режим супервизора.

Виды интерфейсов

Командный (текстовый) интерфейс.

Всякая ОС имеет командный интерфейс (иногда в скрытой форме). Если снять «шелуху» текстовых или графических оболочек и интерфейсов, то «на глубине» всегда можно найти командный интерфейс.

В первых ОС взаимодействие с пользователями было жестко поделено между следующими компонентами:

· командным языком оператора – языком диалогового режима, включающим в себя команды запуска/остановки задач, привязки носителей информации к устройствам, получения информации о заданиях, ожидающих выполнения, вывода, наличии свободной памяти и свободных устройств и др.;

· языком управления заданиями – языком пакетной обработки, на котором прочие пользователи описывали состав и структуру процесса обработки данных: последовательность запуска программ, входные и выходные файлы, условия при которых те или иные программы должны быть выполнены или пропущены и др.

После распространения ПК данное разграничение сошло на нет (в ОС MS-DOS), поскольку пользователь в едином лице соединил функции оператора, администратора и конечного пользователя. Однако, с появлением локальных сетей и более мощных ПК, работающих в многопользовательских режимах, в сетевых ОС и ОС ПК вновь организуется разграничение доступа.

В настоящее время в большинстве ОС сложился более или менее унифицированный формат командной строки, который включает в себя:

· тип операции (имя команды или программы);

· рабочий вход (входные файлы или устройства);

· рабочий выход (выходные файлы или устройства);

· управляющий вход (управляющие параметры или ключи команды)

· управляющий выход (обычно – протокол, содержащий диагностику ошибок, код завершения или другую информацию).