Системное программирование
Л.р.2. Управление процессами

Цель:

Концепция вычислительных процессов и ее реализация в Windows. Основные этапы жизненного цикла процессов и элементарное управление ими: порождение, завершение, получение и изменение состояния.

Теоретическая и методическая часть

Вычислительные процессы. Вычислительный процесс как системный объект, объект Process в Windows. Атрибуты процесса. Образ процесса. Адресное пространство и пространство дескрипторов. Состояния процесса.

Создание (порождение) процессов. Иерархия процессов: родительские (parent) и дочерние (child) процессы.

Завершение процессов. Код завершения.

Программный интерфейс (API) для «базового» управления процессами.

Практическая часть

Общая постановка задачи:

Приложение (или приложения), демонстрирующие простые ситуации, связанные с управлением процессами, в основном их создание, завершение, получение информации о текущем состоянии, а также оценкой влияния на работу системы в целом. (Предполагается, что для выполнения задачи не требуется использовать специализированные механизмы взаимодействия процессов, а также обращаться к подсистеме безопасности.)

Специальных требований к приложениям не предъявляется; в частности, во многих случаях они могут быть не обязательно оконными, но также и консольными.

Варианты заданий:

– Самовосстанавливающийся процесс
– Процесс-«диспетчер»
– Упрощенный аналог планировщика cron
– «Диспетчер зомби»
– ...

1 Самовосстанавливающийся процесс

Возобновление работы процесса с сохранением (продолжением) текущих функций после завершения его сообщением **`WM_CLOSE`**.

Игнорирование сообщения приводит, как правило, либо к объявлению его аварийным (автоматически), либо принудительному безусловному завершению посредством TaskManager.
Решение: процесс завершается штатным образом, но перед этим порождает свою копию (аналогичный процесс из того же исполняемого файла). Копия продолжает выполняться вместо родительского процесса.

Проблема: необходимость продолжить работу с теми же обрабатываемыми данными.
Вариант решения: хранение текущих рабочих данных в структуре («прикладной контекст»), которая может выгружаться на диск или сохраняться в глобальной памяти (родителем) и загружаться оттуда (потомком-«наследником»).

2 Процесс-«диспетчер»

Процесс, выполняющий:
– выбор исполняемого файла, запуск процесса из него
– хранение списка порожденных процессов
– отображение состояния контролируемых процессов (достаточно различать состояния «выполняется», «завершился»)
– возможность послать сообщение **`WM_CLOSE`** выбранному процессу
– отображение возникающих ошибок

В качестве контролируемых процессов можно использовать произвольные подходящие программы либо специально написанный процесс: оконное приложение, способное наглядно показывать свое выполнение.

Проверка состояния процессов – в простейшем случае периодический опрос, например с помощью Wait-функций (WaitForSingleObject(), WaitForMultipleObjects()). При этом проверка не должна блокировать на существенное время выполнение процесса-диспетчера.

3 Упрощенный аналог планировщика cron

Выполнение внешних программ в соответствии с заданным расписанием (конфигурацией), загружаемым из файла/файлов (формат можно заимствовать от cron или использовать свой упрощенный).

Расписание содержит имена исполняемых файлов, время (условие) их выполнения, параметры запуска.

Необходимо обрабатывать возможные ошибки выполнения.

4 «Диспетчер зомби»

Исследование условий появления процессов-«зомби» и способов предотвращения их появления.

Процесс в состоянии «зомби» – после завершения, но до окончательного удаления из системы. «Зомби» не выполняется и не конкурирует за время

процессора, большая часть выделенных для него ресурсов освобождена. Однако информация о соответствующем объекте сохраняется в системных таблицах, что перегружает их и может замедлять работу планировщика.

В Windows процесс остается в этом состоянии до тех пор, пока в системе существует хотя бы один описатель (handle), ссылающийся на этот объект. Следовательно, если родительский процесс закрывает принадлежащий ему handle порожденного процесса и если других открытых handle на этот процесс нет, то процесс после завершения может быть удален окончательно. Но, в то же время, не владея handle своего потомка, процесс-родитель не может управлять им.

Для задания потребуются два процесса:

– процесс-родитель (диспетчер): запуск процесса-потомка, сохраняя или нет его handle (переключатель в интерфейсе), отображение состояния запущенных процессов, закрытие handle завершившихся

– процесс-потомок: демонстрация своего выполнения, завершение по команде (элемент управления), по времени или посредством TaskManager.

Списки и состояния сопоставляются с TaskManager.

5 ...

...