Организация процессов в операционных системах

ЗАДАНИЯ

Для слушателей переподготовки по специальности

"Программное обеспечение информационных систем "

Объект ядра «задание» (job)

- Группу процессов зачастую нужно рассматривать как единую сущность.
- Windows поддерживает объект ядра под названием «задание» (job). Он позволяет группировать процессы и помещать их в нечто вроде песочницы, которая определенным образом ограничивает их действия. Этот объект можно рассматривать как контейнер процессов. Бывает полезно создавать задание и с одним процессом — это позволяет налагать на процесс ограничения, которые иначе указать нельзя.

- void StartRestrictedProcess() {
- // Проверить, не связан ли этот процесс с заданием.
  // Если да, переключиться на другое задание невозможно.
- BOOL bInJob = FALSE;
  IsProcessInJob(GetCurrentProcess(), NULL, &bInJob);
- if (bInJob) { MessageBox(NULL, TEXT("Process already in a job"), TEXT(""), MB_ICONINFORMATION | MB_OK);
  return; }

- // создаем объект ядра "задание"
- HANDLE hjob = CreateJobObject(NULL, TEXT("Wintellect_RestrictedProcessJob"));
- // вводим ограничения для процессов в задании
- // сначала определяем некоторые базовые ограничения
- JOBOBJECT_BASIC_LIMIT_INFORMATION jobli = { 0 };
- // процесс всегда выполняется с классом приоритета idle
- jobli.PriorityClass = IDLE_PRIORITY_CLASS;

- // задание не может использовать более одной секунды процессорного времени
- jobli.PerJobUserTimeLimit.QuadPart = 10000;
- // 1 секунда, выраженная в
- // 100-наносекундных интервалах

- // два ограничения, которые я налагаю на задание (процесс)
- jobli.LimitFlags = J0B_0BJECT_LIMIT_PRI0RITY_CLASS | J0B_0BJECT_LIMIT_J0B_TIME;
- SetInformationJobObject(hjob, JobObjectBasicLimitInformation, &jobli, sizeof(jobli));

- // теперь вводим некоторые ограничения по пользовательскому интерфейсу
- J0B0BJECT_BASIC_UI_RESTRICTI0NS jobuir;
- jobuir.UIRestrictionsClass = J0B_0BJECT_UILIMIT_N0NE;
- // "замысловатый" нуль
- // процесс не имеет права останавливать систему jobuir.UIRestrictionsClass |= J0B_0BJECT_UILIMIT_EXITWIND0WS;
- // Процесс не имеет права обращаться к USER-объектам в системе
- // (например, к другим окнам).
- jobuir.UIRestrictionsClass |= J0B_0BJECT_UILIMIT_HANDLES;
- SetInformationJobObject(hjob, JobObjectBasicUIRestrictions, &jobuir, sizeof(jobuir));

```
// Порождаем процесс, который будет размещен в задании.
// ПРИМЕЧАНИЕ: процесс нужно сначала создать и только потом
// поместить в задание. А это значит, что поток процесса должен быть
// создан  и тут же приостановлен, чтобы он не смог выполнить какой-
// нибудь код еще до введения ограничений.
STARTUPINFO si = { sizeof(si) };
PROCESS_INFORMATION pi;
TCHAR szCmdLine[8];
 _tcscpy_s(szCmdLine, _countof(szCmdLine), TEXT("CMD"));
BOOL bResult = CreateProcess( NULL, szCmdLine, NULL, NULL, FALSE,
CREATE_SUSPENDED | CREATE_NEW_CONSOLE, NULL, NULL, &si, &pi);
```

- // Включаем процесс в задание.
- // ПРИМЕЧАНИЕ: дочерние процессы, порождаемые этим
- // процессом, автоматически становятся частью того же
- // задания.
- AssignProcessToJobObject(hjob, pi.hProcess);
- // теперь потоки дочерних процессов могут выполнять код ResumeThread(pi.hThread);
- CloseHandle(pi.hThread);

```
// Ждем, когда процесс завершится или будет исчерпан.
// Лимит процессорного времени, указанный для задания.

HANDLE h[2]; h[0] = pi.hProcess;
h[1] = hjob;
DWORD dw = WaitForMultiple0bjects(2, h, FALSE, INFINITE);
switch (dw - WAIT_OBJECT_0)
{
    case 0: // процесс завершился
            …
            break;
    case 1: // лимит процессорного времени исчерпан
            …
            break;
}
```

- FILETIME CreationTime; FILETIME ExitTime;
- FILETIME KernelTime; FILETIME UserTime;
- TCHAR szInfo[MAX_PATH];
- GetProcessTimes(pi.hProcess, ACreationTime, &ExitTime, &KernelTime, &UserTime);
- StringCchPrintf(szInfo, _countof(szInfo), TEXT("Kernel = %u | User = %u\n"), KernelTime.dwLowDateTime / 10000, UserTime.dwLowDateTime / 10000);
- MessageBox(GetActiveWindow(), szInfo, TEXT("Restricted Process times"), MB_ICONINFORMATION | MB_OK);
- // проводим очистку
- CloseHandle(pi.hProcess); CloseHandle(hjob);
- }

Определение ограничений, налагаемых на процессы в задании

- Ограничения бывают нескольких видов:
- ■ базовые и расширенные базовые ограничения — не дают процессам в задании моно- польно захватывать системные ресурсы;
- ■ базовые ограничения по пользовательскому интерфейсу (UI) — блокируют возмож- ность его изменения;
- ■ ограничения, связанные с защитой, — перекрывают процессам в задании доступ к за- щищенным ресурсам (файлам, подразделам реестра и т. д.).

Ограничения на задание вводятся вызовом:

- **BOOL SetInformationJobObject(HANDLE hJob, JOBOBJECTINFOCLASS JobObjectInformationClass, PVOID pJobObjectInformation,**
- **DWORD cbJobObjectInformationSize);**

- typedef struct _JOBOBJECT_BASIC_LIMIT_INFORMATION {
  - LARGE_INTEGER PerProcessUserTimeLimit;
  - LARGE_INTEGER PerJobUse rTimeLimit;
  - DWORD LimitFlags;
  - DWORD MinimumWorkingSetSize;
  - DWORD MaximumWorkingSetSize;
  - DWORD ActiveProcessLimit;
  - DWORD_PTR Affinity;
  - DWORD PriorityClass;
  - DWORD SchedulingClass;
- } JOBOBJECT_BASIC_LIMIT_INFORMATION, *PJOBOBJECT_BASIC_LIMIT_INFORMATION;

- **typedef struct _JOBOBJECT_BASIC_UI_RESTRICTIONS {**
  - **DWORD UIRestrictionsClass;**
- **} JOBOBJECT_BASIC_UI_RESTRICTIONS, *PJOBOBJECT_BASIC_UI_RESTRICTIONS;**
  - JOB_OBJECT_UILIMIT_ EXITWINDOWS
  - JOB_OBJECT_UILIMIT_ READCLIPBOARD
  - JOB_OBJECT_UILIMIT_ WRITECLIPBOARD
  - JOB_OBJECT_UILIMIT_ SYSTEMPARAMETERS
  - JOB_OBJECT_UILIMIT_ DISPLAYSETTINGS
  - JOB_OBJECT_UILIMIT_ GLOBALATOMS
  - JOB_OBJECT_UILIMIT_ DESKTOP
  - JOB_OBJECT_UILIMIT_ HANDLES

- typedef struct _JOBOBJECT_SECURITY_LIMIT_INFORMATION {
  - DWORD SecurityLimitFlags; HANDLE JobToken;
  - PTOKEN_GROUPS SidsToDisable;
  - PTOKEN_PRIVILEGES PrivilegesToDelete;
  - PTOKEN_GROUPS RestrictedSids;
- } JOBOBJECT_SECURITY_LIMIT_INFORMATION, *PJOBOBJECT_SECURITY_LIMIT_INFORMATION;

- **BOOL QueryInformationJobObject(HANDLE hJob, JOBOBJECTINFOCLASS JobObjectInformationClass, PVOID pvJobObjectInformation, DWORD cbJobObjectIInformationSize, PDWORD pdwReturnSize);**

Включение процесса в задание

- **BOOL AssignProcessToJobObject(HANDLE hJob, HANDLE hProcess);**

Завершение всех процессов в задании

- **BOOL TerminateJobObject(HANDLE hJob, UINT uExitCode);**

Получение статистической информации о задании

- *QueryInformationjobObject*

- Например, чтобы выяснить базовые учетные сведения, вызовите ее, передав *JobObjectBasicAccountingInformation* во втором параметре и адрес структуры
- JOBOBJECT_ BASIC_ACCOUNTING_INFORMATION

- typedef struct _JOBOBJECT_BASIC_ACCOUNTING_INFORMATION { LARGE_INTEGER TotalUserTime; LARGE_INTEGER TotalKernelTime; LARGE_INTEGER ThisPeriodTotalUserTime; LARGE_INTEGER ThisPeriodTotalKernelTime; DWORD TotalPageFaultCount; DWORD TotalProcesses; DWORD ActiveProcesses; DWORD TotalTerminatedProcesses; } JOBOBJECT_BASIC_ACCOUNTING_INFORMATION, *PJOBOBJECT_BASIC_ACCOUNTING_INFORMATION;