Университет ИТМО

Кафедра ИПМ

Системное программное обеспечение

Лабораторная работа № 3

Управление процессами

Работу выполнил:

Студент 2 курса

Группы № 2125

Назарьев Сергей

Санкт-Петербург

2014 г.

**Цель работы:**

1. Расширьте возможности программы 6.1 (grepMP) таким образом, чтобы она принимала также параметры командной строки, а не только текстовый шаблон.
2. Вместо того чтобы передавать дочернему процессу имя временного файла, как это делается в программе 6.1, преобразуйте наследуемый дескриптор файла к типу DWORD (для типа HANDLE требуется 4 байта), а затем в строку символов. Передайте эту строку дочернему процессу в командной строке. В свою очередь, дочерний процесс должен осуществить обратное преобразование строки символов в значение дескриптора файла, который будет использован для вывода. Эту методику иллюстрируют программы catHA.с и grepHA.с, доступные на Web-сайте книги.
3. Программа 6.1 ожидает завершения всех процессов и лишь после этого выводит результаты. При этом возможность определения того, в каком именно порядке завершились процессы внутри программы, отсутствует. Модифицируйте программу таким образом, чтобы она определяла очередность завершения процессов. Подсказка. Измените вызов функции WaitForMultipleObjects таким образом, чтобы возврат из нее осуществлялся после завершения каждого отдельного процесса. Другой возможностью является сортировка времени завершения процессов.

**Код программы:**

#include "Everything.h"

**char** patternArg = -**1**;

**char** ignoreCase = **0**;

**char** printed = **0**;

**int** **\_tmain** (**int** argc, LPTSTR argv[])

{

HANDLE hTempFile;

SECURITY\_ATTRIBUTES stdOutSA = {**sizeof** (SECURITY\_ATTRIBUTES), NULL, TRUE};

TCHAR commandLine[MAX\_PATH + **100**];

STARTUPINFO startUpSearch, startUp;

PROCESS\_INFORMATION processInfo;

DWORD exitCode, dwCreationFlags = **0**;

**int** iProc = **0**;

HANDLE \*hProc;

HANDLE \*hProc2;

HANDLE \*hProc3;

**typedef** **struct** {TCHAR tempFile[MAX\_PATH];} PROCFILE;

PROCFILE \*procFile;

#ifdef UNICODE

dwCreationFlags = CREATE\_UNICODE\_ENVIRONMENT;

#endif

**if** (argc < **3**) {

fprintf (stderr, \_T ("Usage: grepMP [-y] pattern files."));

exit(**1**);

}

GetStartupInfo (&startUpSearch);

GetStartupInfo (&startUp);

procFile = malloc (argc \* **sizeof** (PROCFILE));

hProc = malloc (argc \* **sizeof** (HANDLE));

hProc2 = malloc (argc \* **sizeof** (HANDLE));

hProc3 = malloc (argc \* **sizeof** (HANDLE));

**int** i;

**for** (i = **1**; i < argc; ++i) {

**if** (i < argc && argv [i] [**0**] == '-') {

**switch** (argv [i] [**1**]) {

**case** 'y':

ignoreCase = **1**;

**break**;

}

} **else** **if**(patternArg == -**1**) {

patternArg = i;

} **else** {

**if**(!printed) printed = i;

**if** (GetTempFileName (\_T ("."), \_T ("gtm"), **0**, procFile[iProc].tempFile) == **0**)

fprintf (stderr, \_T ("Temp file failure."));

hTempFile = CreateFile(procFile[iProc].tempFile,

GENERIC\_WRITE,

FILE\_SHARE\_READ | FILE\_SHARE\_WRITE, &stdOutSA,

CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL

);

**if** (hTempFile == INVALID\_HANDLE\_VALUE)

fprintf (stderr, \_T ("Failure opening temp file."));

\_stprintf (commandLine, \_T ("grep %s**\"**%u**\"** **\"**%s**\"** **\"**%s**\"**"),

(ignoreCase ? "-y " : ""),

(DWORD)hTempFile,

argv[patternArg],

argv[i]

);

startUpSearch.dwFlags = STARTF\_USESTDHANDLES;

**if** (!CreateProcess (NULL, commandLine, NULL, NULL,

TRUE, dwCreationFlags, NULL, NULL, &startUpSearch, &processInfo))

fprintf (stderr, \_T ("ProcCreate failed."));

CloseHandle (hTempFile); CloseHandle (processInfo.hThread);

hProc[iProc++] = processInfo.hProcess;

}

}

**int** sizeOfProc = iProc;

**int** sz;

**int** j;

**int** a = **0**;

**int** realId;

**for**(j = **0**; j < sizeOfProc; j++)

hProc2[a++] = hProc[j];

**for** (iProc = **0**; iProc < sizeOfProc; iProc += MAXIMUM\_WAIT\_OBJECTS) {

sz = min(MAXIMUM\_WAIT\_OBJECTS, sizeOfProc);

**while**(TRUE) {

DWORD id = WaitForMultipleObjects (sz,

&hProc2[iProc], FALSE, INFINITE) - WAIT\_OBJECT\_0 + iProc;

**for**(j = **0**; j < sizeOfProc; j++) {

**if**(hProc[j] == hProc2[id]) {

realId = j;

**break**;

}

}

**if** (GetExitCodeProcess (hProc2[id], &exitCode) && exitCode == **0**) {

\_tprintf (\_T("%s:**\n**"), argv[realId + printed]);

\_stprintf (commandLine, \_T ("cat **\"**%s**\"**"), procFile[realId].tempFile);

**if** (!CreateProcess (NULL, commandLine, NULL, NULL,

TRUE, dwCreationFlags, NULL, NULL, &startUp, &processInfo))

fprintf (stderr, \_T ("Failure executing cat."));

**else** {

WaitForSingleObject (processInfo.hProcess, INFINITE);

CloseHandle (processInfo.hProcess);

CloseHandle (processInfo.hThread);

}

}

CloseHandle (hProc2[id]);

a = **0**;

**for**(j = **0**; j < sz; j++)

**if**(j != id)

hProc3[a++] = hProc2[iProc+j];

sz--;

**for**(j = **0**; j < sz; j++)

hProc2[iProc+j] = hProc3[j];

**if** (!DeleteFile (procFile[realId].tempFile))

fprintf (stderr, \_T ("Cannot delete temp file.**\n**"));

**if**(sz == **0**) **break**;

}

}

**if** (!printed) {

fprintf (stderr, \_T ("Usage: grepMP [-y] pattern files."));

exit(**1**);

}

**return** **0**;

}

**Вывод:**