ИТМО Кафедра Информатики и прикладной математики

Отчет по лабораторной работе №4 «Управление процессами»

Выполнил: студент группы P3217
Плюхин Дмитрий
Преподаватель: Зыков А. Г.

2017 год

1. Задание

Разработать алгоритм и программу, реализующую следующие действия:
Сгенерировать (или использовать из предыдущих заданий) массив данных, который будет использоваться для сортировки и поиска в нём элементов.
Создать три отдельных процесса, в каждом из которых:
a) Отсортировать исходный массив по заданному в командной строке шаблону.
b) Осуществить поиск элемента, указанного также в командной строке, в отсортированном массиве.
Вывести временные характеристики и ID процессов.
Определить количество открытых дескрипторов.

2. Листинг основной части программы

Файл main.cpp содержит функцию, получающую на вход четыре строковые константы : имя файла для сортировки, шаблон сортировки, имя файла для поиска, строка для поиска. Функция позволяет отсортировать предложения в текстовом файле, а затем выполнить поиск по какому-либо шаблону и вывести на печать предложения в отсортированном порядке, содержащие шаблон, по которому производится поиск. Далее приведены ключевые функции из файла main.cpp.

```
int sort(const char* fileName, const char* sortTemplate){
  char sortedFileName[PATH_MAX_LEN];
  strncpy(sortedFileName,fileName,PATH_MAX_LEN);
  strcat(sortedFileName,".srtd");

  CopyFile(fileName, sortedFileName, FALSE);
  HANDLE hFile = CreateFile(fileName,GENERIC_READ | GENERIC_WRITE, 0, NULL, OPEN_EXISTING, 0,
NULL);
  DWORD fileSize = GetFileSize(hFile,NULL);
  HANDLE hMap = CreateFileMapping(hFile, NULL, PAGE_READWRITE, 0, fileSize, NULL);
  char* mappedFile = (char*)MapViewOfFile(hMap, FILE_MAP_ALL_ACCESS, 0, 0, fileSize);
  //replaceSpaces(mappedFile,fileSize);
  char* mappedFileNull = mappedFile;
  HANDLE hSentencesHeap = HeapCreate(HEAP_GENERATE_EXCEPTIONS | HEAP_NO_SERIALIZE,
HEAP_SIZE_DEFAULT, 0);

  DWORD numOfSentences =  getNumOfSentences(mappedFile, fileSize);
  PSENTENCE sentences = (PSENTENCE)HeapAlloc(hSentencesHeap, HEAP_ZERO_MEMORY,
numOfSentences*sizeof(SENTENCE));
  DWORD index = 0;
  DWORD accumulator = 0;
  for (int i = 0; i < numOfSentences; i++){
    (sentences+i)->pointer = mappedFile + accumulator;
    (sentences+i)->length = getSentenceLength(mappedFile + accumulator);
    accumulator += (sentences+i)->length;
  }

  if (strcmp(sortTemplate,"asc")==0){
    qsort(sentences, numOfSentences, sizeof(SENTENCE), Compare);
  } else if (strcmp(sortTemplate,"desc")==0){
    qsort(sentences, numOfSentences, sizeof(SENTENCE), CompareDesc);
  }


  HANDLE hFileS = CreateFile(sortedFileName,GENERIC_READ | GENERIC_WRITE, 0, NULL,
OPEN_EXISTING, 0, NULL);
  DWORD fileSizeS = GetFileSize(hFileS,NULL);
  HANDLE hMapS = CreateFileMapping(hFileS, NULL, PAGE_READWRITE, 0, fileSizeS, NULL);
  char* mappedFileS = (char*)MapViewOfFile(hMapS, FILE_MAP_ALL_ACCESS, 0, 0, fileSizeS);
  char* mappedFileNullS = mappedFileS;

  accumulator = 0;
  for (int i = 0; i < numOfSentences; i++){
    strncpy(mappedFileS+accumulator,(sentences+i)->pointer,(sentences+i)->length);
    accumulator += (sentences+i)->length;
  }

  UnmapViewOfFile(mappedFileS);
  CloseHandle(hMapS);
```

```
  CloseHandle(hFileS);

  UnmapViewOfFile(mappedFile);
  CloseHandle(hMap);
  CloseHandle(hFile);

  return 0;
}

int search(const char* fileName, const char* searched){
  char tmpFileName[PATH_MAX_LEN];
  strncpy(tmpFileName,fileName,PATH_MAX_LEN);
  strcat(tmpFileName,".tmp");
  DWORD searchedLen = strlen(searched);

  CopyFile(fileName, tmpFileName, FALSE);

  HANDLE hFile = CreateFile(tmpFileName,GENERIC_READ | GENERIC_WRITE, 0, NULL, OPEN_EXISTING, 0,
NULL);
  DWORD fileSize = GetFileSize(hFile,NULL);
  HANDLE hMap = CreateFileMapping(hFile, NULL, PAGE_READWRITE, 0, fileSize, NULL);
  char* mappedFile = (char*)MapViewOfFile(hMap, FILE_MAP_ALL_ACCESS, 0, 0, fileSize);
  char* mappedFileNull = mappedFile;

  mappedFile[fileSize - 1] = 0;
  for (DWORD i = 0; i < fileSize; i++){
    if (check(mappedFile+i,searched)){
      showSentence(mappedFile+i,fileSize-i-1,i);
      i += searchedLen - 1;
    }
  }
  UnmapViewOfFile(mappedFile);
  CloseHandle(hMap);
  CloseHandle(hFile);
  return 0;
}

int main(int argc, char* argv[]){
  setlocale(LC_ALL,"Russian");
  if (argc != 5){
    cout << "Usage : main name_of_file_for_sorting template_for_sorting
name_of_file_for_searching str_for_searching" << endl;
    return 1;
  }
  sort(argv[1],argv[2]);
  search(argv[3],argv[4]);
}
```

Файл launcher.cpp содержит код запуска трех процессов для сортировки внешнего файла, поиска в отсортированном файле и вывода на печать ID процессов и их временных характеристик по завершении их выполнения, а также количество открытых дескрипторов.

```
int showTime(const char* overture, LPSYSTEMTIME systemTime){
  cout << overture << systemTime->wHour << ":" << systemTime->wMinute << ":" << systemTime-
>wSecond << ":" << systemTime->wMilliseconds;
  return 0;
}


int main(int argc, char* argv[]){
  setlocale(LC_ALL,"Russian");
  HANDLE* processHandles;
  DWORD* processIds;
  DWORD numOfOpenedHandles;
  processHandles = (HANDLE*)malloc(3*sizeof(HANDLE));
  processIds = (DWORD*)malloc(3*sizeof(DWORD));

  STARTUPINFO startUpInfo;
  FILETIME creationFileTime;
  SYSTEMTIME creationSystemTime;
  FILETIME exitFileTime;
  SYSTEMTIME exitSystemTime;
  FILETIME userFileTime;
  SYSTEMTIME userSystemTime;
  FILETIME kernelFileTime;
  SYSTEMTIME kernelSystemTime;
```

```
  SECURITY_ATTRIBUTES StdOutSA ={sizeof(SECURITY_ATTRIBUTES), NULL, TRUE};

  HANDLE hFileLog = CreateFile("log.out",GENERIC_WRITE, FILE_SHARE_READ | FILE_SHARE_WRITE,
&StdOutSA, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);


  PROCESS_INFORMATION processInformationMorning;
  PROCESS_INFORMATION processInformationEvening;
  PROCESS_INFORMATION processInformationDay;
  const char* commandLineMorning = "main data/lullaby.txt asc data/lullaby.txt.srtd утро";
  const char* commandLineDay = "main data/ghosts.txt desc data/ghosts.txt.srtd ночь";
  const char* commandLineEvening = "main data/widow.txt asc data/widow.txt.srtd вечер";
  GetStartupInfo(&startUpInfo);
  startUpInfo.dwFlags = STARTF_USESTDHANDLES;
  startUpInfo.hStdOutput = hFileLog;

  CreateProcess(NULL, (LPSTR)commandLineMorning, NULL, NULL, TRUE, 0, NULL, NULL, &startUpInfo,
&processInformationMorning);
  CreateProcess(NULL, (LPSTR)commandLineDay, NULL, NULL, TRUE, 0, NULL, NULL, &startUpInfo,
&processInformationDay);
  CreateProcess(NULL, (LPSTR)commandLineEvening, NULL, NULL, TRUE, 0, NULL, NULL, &startUpInfo,
&processInformationEvening);

  processHandles[0] = processInformationMorning.hProcess;
  processHandles[1] = processInformationDay.hProcess;
  processHandles[2] = processInformationEvening.hProcess;

  processIds[0] = processInformationMorning.dwProcessId;
  processIds[1] = processInformationDay.dwProcessId;
  processIds[2] = processInformationEvening.dwProcessId;

  WaitForMultipleObjects(3,processHandles,TRUE,INFINITE);
  for (int i = 0; i < 3; i++){
    cout << "Process " << processIds[i];

GetProcessTimes(processHandles[i],&creationFileTime,&exitFileTime,&kernelFileTime,&userFileTime)
;
    FileTimeToSystemTime(&creationFileTime,&creationSystemTime);
    FileTimeToSystemTime(&exitFileTime,&exitSystemTime);
    FileTimeToSystemTime(&userFileTime,&userSystemTime);
    FileTimeToSystemTime(&kernelFileTime,&kernelSystemTime);
    cout << showTime(" was created at ",&creationSystemTime);
    cout << showTime(", was exited at ",&exitSystemTime);
    cout << showTime(", so it was handling by kernel in ",&kernelSystemTime);
    cout << showTime(", and by user in ",&userSystemTime) << endl;
       GetProcessHandleCount(processHandles[i],&numOfOpenedHandles);
       cout << "In the process " << numOfOpenedHandles << " descriptors were opened" << endl;
  }
  CloseHandle(hFileLog);
}
```

3. Результаты работы программы

В результате работы программы получены следующие числовые характеристики:

```
Process 3512 was created at 10:58:11:4300, was exited at 10:58:12:1800, so it was handling by kernel in 0:0:0:460, and by user in 0:0:0:460
In the process 0 descriptors were opened
Process 11532 was created at 10:58:11:4380, was exited at 10:58:11:5680, so it was handling by kernel in 0:0:0:310, and by user in 0:0:0:620
In the process 0 descriptors were opened
Process 8352 was created at 10:58:11:4450, was exited at 10:58:11:5790, so it was handling by kernel in 0:0:0:930, and by user in 0:0:0:310
In the process 0 descriptors were opened
```

4. Вывод

Таким образом, механизм управления процессами в среде Windows не представляет особенной сложности: процесс создается при помощи функции CreateProcess, множество параметров которой обеспечивают исключительную гибкость, например, можно произвольно установить поток ввода или вывода, а также настроить параметры командной строки. Функция GetProcessTimes позволяет получить временные характеристики процесса, однако неудобство составляет необходимость ручной конвертации возвращаемых данных в более удобный для манипуляции формат SYSTEMTIME.