Московский Авиационный Институт

(Национальный Исследовательский Университет)

Кафедра: 806 «Вычислительная математика и программирование»

Факультет: «Прикладная математика и физика»

Дисциплина: «Операционные системы»

Лабораторная работа №4.

Группа: 8О-204Б

Студент: Лебедев Т.А.

Вариант: №

Оценка:

Подпись:

Москва

2016

#define \_CRT\_SECURE\_NO\_WARNINGS

#ifndef MAPPING\_H\_

#define MAPPING\_H\_

#include <windows.h>

#include <stdio.h>

#include <string.h>

#include <ctype.h>

typedef struct {

TCHAR\* nameFile;

HANDLE hFile;

HANDLE hFileMap;

DWORD64 sizeFile;

DWORD sizeMap;

DWORD sizeMemory;

} TMapping;

BOOL CreateFileMap(TMapping \*map, const TCHAR\* nameFile, DWORD sizeMemory, DWORD64 minSizeFile, DWORD64 maxSizeFile);

BOOL CloseFileMap(TMapping \*map);

DWORD64 FileSize(HANDLE hFile);

DWORD64 FileNumLine(TMapping \*map, DWORD sysGranularity);

BOOL GetLine(TMapping\* map, DWORD64 numLine, DWORD sysGranularity);

DWORD64 Search(TMapping\* map, DWORD sizeSample, TCHAR\* sample, BOOL reg, DWORD sysGranularity);

BOOL Replace(TMapping\* map, DWORD sizeSample, TCHAR\* sample, DWORD sizeRepl, TCHAR\* repl,

DWORD sysGranularity, DWORD64 minSizeFile, DWORD64 maxSizeFile);

#endif

#include "mapping.h"

int CreateFileMap(TMapping \*map, const TCHAR\* nameFile, DWORD sizeMemory, DWORD64 minSizeFile, DWORD64 maxSizeFile){

if (minSizeFile > maxSizeFile) {

printf("ERROR: wrong file size limit\n ");

return FALSE;

}

map->sizeMemory = sizeMemory;

map->hFile = CreateFile(nameFile, GENERIC\_READ, 0, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL);

if (map->hFile == INVALID\_HANDLE\_VALUE) {

printf("ERROR: can't open file\n");

map->hFile = NULL;

map->hFileMap = NULL;

return FALSE;

}

map->sizeFile = FileSize(map->hFile);

if (!map->sizeFile) {

CloseHandle(map->hFile);

map->hFile = NULL;

map->hFileMap = NULL;

printf("ERROR: empty file\n");

return FALSE;

}

if (minSizeFile > map->sizeFile || maxSizeFile < map->sizeFile) {

printf("the file does not meet the limits\n");

CloseHandle(map->hFile);

map->hFile = NULL;

map->hFileMap = NULL;

return FALSE;

}

map->hFileMap = CreateFileMapping(map->hFile, NULL, PAGE\_READONLY, 0, 0, NULL);

if (map->hFileMap == NULL) {

CloseHandle(map->hFile);

printf("ERROR: create mapping\n");

map->hFile = NULL;

map->hFileMap = NULL;

return FALSE;

}

int size = strlen(nameFile);

map->nameFile = malloc(sizeof(char)\* (size + 1));

strcpy(map->nameFile, nameFile);

return TRUE;

}

BOOL CloseFileMap(TMapping\* map) {

BOOL f;

BOOL ans = TRUE;

f = CloseHandle(map->hFileMap);

if (!f) {

printf("ERROR: close map\n");

ans = FALSE;

}

f = CloseHandle(map->hFile);

if (!f) {

printf("ERROR: close file\n");

ans = FALSE;

}

map->hFile = NULL;

map->hFileMap = NULL;

return ans;

}

DWORD64 FileSize(HANDLE hFile) {

DWORD hight, lower;

lower = GetFileSize(hFile, &hight);

return (DWORD64)hight << 32 | lower;

}

DWORD64 FileNumLine(TMapping \*map, DWORD sysGranularity) {

if (map == NULL) {

printf("ERROR:map not create\n");

return FALSE;

}

DWORD64 indexLine = 1;

DWORD64 indexRead = 0;

DWORD startMapping;

DWORD sizeView;

BOOL notEnd = TRUE;

TCHAR\* readingInMap;

LPVOID pointerMap;

while (notEnd) {

startMapping = (indexRead / sysGranularity) \* sysGranularity;

sizeView = (indexRead % sysGranularity) + ((indexRead + map->sizeMemory) < map->sizeFile ? map->sizeMemory : (map->sizeFile - indexRead));

pointerMap = MapViewOfFile(map->hFileMap, FILE\_MAP\_READ, (DWORD)(indexRead >> 0x20), startMapping, sizeView);

if (pointerMap == NULL) {

printf("ERROR:unmap view\n");

return 0;

}

int distanceToStartRead = indexRead - startMapping;

readingInMap = (TCHAR\*)pointerMap + distanceToStartRead;

int sizeReadingInMap = sizeView - distanceToStartRead;

for (int i = 0; i < sizeReadingInMap; i++) {

if (readingInMap[i] == '\r' && readingInMap[i + 1] == '\n'){

i++;

indexLine++;

}

}

if ((startMapping + sizeView) == map->sizeFile)

notEnd = FALSE;

if (!UnmapViewOfFile(pointerMap)) {

printf("ERROR: unmap view\n");

return 0;

}

indexRead += sizeReadingInMap;

}

return indexLine;

}

BOOL GetLine(TMapping\* map, DWORD64 numLine, DWORD sysGranularity) {

if (map == NULL) {

printf("ERROR:map not create\n");

return FALSE;

}

DWORD64 indexLine = 1;

DWORD64 indexRead = 0;

DWORD startMapping;

DWORD sizeView;

BOOL notEnd = TRUE;

TCHAR\* readingInMap;

LPVOID pointerMap;

while (notEnd) {

startMapping = (indexRead / sysGranularity) \* sysGranularity;

sizeView = (indexRead % sysGranularity) + ((indexRead + map->sizeMemory) < map->sizeFile ? map->sizeMemory : (map->sizeFile - indexRead));

pointerMap = MapViewOfFile(map->hFileMap, FILE\_MAP\_READ, (DWORD)(indexRead >> 0x20), startMapping, sizeView);

if (pointerMap == NULL) {

printf("ERROR:unmap view\n");

return FALSE;

}

int distanceToStartRead = indexRead - startMapping;

readingInMap = (TCHAR\*)pointerMap + distanceToStartRead;

int sizeReadingInMap = sizeView - distanceToStartRead;

for (int i = 0; i < sizeReadingInMap; i++) {

if (indexLine == numLine)

printf("%c", readingInMap[i]);

if (readingInMap[i] == '\r' && readingInMap[i + 1] == '\n'){

i++;

indexLine++;

}

if (indexLine > numLine) {

notEnd = FALSE;

break;

}

}

if ((startMapping + sizeView) == map->sizeFile)

notEnd = FALSE;

if (!UnmapViewOfFile(pointerMap)) {

printf("ERROR: unmap view\n");

return FALSE;

}

indexRead += sizeReadingInMap;

}

return TRUE;

}

DWORD64 Search(TMapping\* map, DWORD sizeSample, TCHAR\* sample, BOOL reg, DWORD sysGranularity) {

if (map == NULL) {

printf("ERROR:map not create\n");

return -1;

}

if (sizeSample == 0) {

return -2;

}

DWORD64 indexRead = 0;

DWORD startMapping;

DWORD sizeView;

BOOL notEnd;

BOOL yes = FALSE;

TCHAR\* readingInMap;

LPVOID pointerMap;

int indexSample;

DWORD64 i;

DWORD64 num = -2;

for (i = 0; i < (map->sizeFile - sizeSample + 1); i++) {

indexSample = 0;

indexRead = i;

notEnd = TRUE;

while (notEnd) {

startMapping = (indexRead / sysGranularity) \* sysGranularity;

sizeView = (indexRead % sysGranularity) + ((indexRead + map->sizeMemory) < map->sizeFile ? map->sizeMemory : (map->sizeFile - indexRead));

pointerMap = MapViewOfFile(map->hFileMap, FILE\_MAP\_READ, (DWORD)(indexRead >> 0x20), startMapping, sizeView);

if (pointerMap == NULL) {

printf("ERROR:map view\n");

return -1;

}

int distanceToStartRead = indexRead - startMapping;

readingInMap = (TCHAR\*)pointerMap + distanceToStartRead;

int sizeReadingInMap = sizeView - distanceToStartRead;

for (int i = 0; i < sizeReadingInMap && indexSample < sizeSample; i++) {

if (reg) {

if (readingInMap[i] != sample[indexSample]) {

notEnd = FALSE;

break;

}

}

else {

if ((char)tolower(readingInMap[i]) != (char)tolower(sample[indexSample])) {

notEnd = FALSE;

break;

}

}

indexSample++;

}

if (indexSample == sizeSample) {

yes = TRUE;

num = i;

notEnd = FALSE;

}

if ((startMapping + sizeView) == map->sizeFile)

notEnd = FALSE;

if (!UnmapViewOfFile(pointerMap)) {

printf("ERROR: unmap view\n");

return -1;

}

indexRead += sizeReadingInMap;

}

if (yes)

break;

}

return num;

}

BOOL Replace(TMapping\* map, DWORD sizeSample, TCHAR\* sample, DWORD sizeRepl, TCHAR\* repl,

DWORD sysGranularity, DWORD64 minSizeFile, DWORD64 maxSizeFile) {

DWORD64 startRepl = Search(map, sizeSample, sample, FALSE, sysGranularity);

if (startRepl == -2) {

printf("Sample not found in file\n");

return TRUE;

}

TMapping\* mapRepl = malloc(sizeof(TMapping));

TCHAR nameFileRepl[10] = TEXT("~2589");

mapRepl->hFile = CreateFile(nameFileRepl, GENERIC\_READ | GENERIC\_WRITE, 0, NULL, OPEN\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);

if (mapRepl->hFile == INVALID\_HANDLE\_VALUE) {

printf("ERROR: can't create file\n");

return FALSE;

}

mapRepl->sizeMemory = map->sizeMemory;

mapRepl->sizeFile = map->sizeFile - sizeSample + sizeRepl;

mapRepl->nameFile = nameFileRepl;

if (!mapRepl->sizeFile) {

if (!CloseFileMap(map))

return FALSE;

if (!CloseHandle(mapRepl->hFile)) {

printf("ERROR: close file\n");

return FALSE;

}

DeleteFile(map->nameFile);

MoveFile(mapRepl->nameFile, map->nameFile);

printf("close file (file empty)\n");

map->sizeFile = mapRepl->sizeFile;

free(mapRepl);

return TRUE;

}

if (minSizeFile > mapRepl->sizeFile || maxSizeFile < mapRepl->sizeFile) {

printf("ERROR: after replacing the file will not comply with limits\n ");

return TRUE;

}

mapRepl->hFileMap = CreateFileMapping(mapRepl->hFile, NULL, PAGE\_READWRITE, (DWORD)(mapRepl->sizeFile >> 0x20), (DWORD)mapRepl->sizeFile, NULL);

if (mapRepl->hFileMap == NULL) {

printf("ERROR: create mapping\n");

return FALSE;

}

mapRepl->sizeMemory = map->sizeMemory;

DWORD64 indexRead1 = 0;

DWORD64 indexRead2 = 0;

DWORD startMapping1;

DWORD startMapping2;

DWORD sizeView1;

DWORD sizeView2;

BOOL notEnd = TRUE;

TCHAR\* readingInMap1;

TCHAR\* readingInMap2;

LPVOID pointerMap1;

LPVOID pointerMap2;

while (notEnd) {

startMapping1 = (indexRead1 / sysGranularity) \* sysGranularity;

sizeView1 = (indexRead1 % sysGranularity) + ((indexRead1 + map->sizeMemory) < map->sizeFile ? map->sizeMemory : (map->sizeFile - indexRead1));

pointerMap1 = MapViewOfFile(map->hFileMap, FILE\_MAP\_READ, (DWORD)(indexRead1 >> 0x20), startMapping1, sizeView1);

if (pointerMap1 == NULL) {

printf("ERROR:map view \n");

return FALSE;

}

int distanceToStartRead1 = indexRead1 - startMapping1;

readingInMap1 = (TCHAR\*)pointerMap1 + distanceToStartRead1;

int sizeReadingInMap = sizeView1 - distanceToStartRead1;

startMapping2 = (indexRead2 / sysGranularity) \* sysGranularity;

sizeView2 = (indexRead2 % sysGranularity) + ((indexRead2 + mapRepl->sizeMemory) < mapRepl->sizeFile ? mapRepl->sizeMemory : (mapRepl->sizeFile - indexRead2));

pointerMap2 = MapViewOfFile(mapRepl->hFileMap, FILE\_MAP\_WRITE, (DWORD)(indexRead2 >> 0x20), startMapping2, sizeView2);

if (pointerMap2 == NULL) {

printf("ERROR:map view\n");

return FALSE;

}

int distanceToStartRead2 = indexRead2 - startMapping2;

readingInMap2 = (TCHAR\*)pointerMap2 + distanceToStartRead2;

for (int i = 0; i < sizeReadingInMap; i++) {

if ((indexRead1 + i) == startRepl) {

notEnd = FALSE;

break;

}

readingInMap2[i] = readingInMap1[i];

}

/\* if ((startMapping2 + sizeView2) == map->sizeFile)

notEnd = FALSE;\*/

if (!UnmapViewOfFile(pointerMap1)) {

printf("ERROR: unmap view\n");

return FALSE;

}

if (!UnmapViewOfFile(pointerMap2)) {

printf("ERROR: unmap view\n");

return FALSE;

}

indexRead1 += sizeReadingInMap;

indexRead2 += sizeReadingInMap;

}

indexRead2 = startRepl;

notEnd = TRUE;

int indexInRepl = 0;

while (notEnd) {

startMapping2 = (indexRead2 / sysGranularity) \* sysGranularity;

sizeView2 = (indexRead2 % sysGranularity) + ((indexRead2 + mapRepl->sizeMemory) < mapRepl->sizeFile ? mapRepl->sizeMemory : (mapRepl->sizeFile - indexRead2));

pointerMap2 = MapViewOfFile(mapRepl->hFileMap, FILE\_MAP\_WRITE, (DWORD)(indexRead2 >> 0x20), startMapping2, sizeView2);

if (pointerMap2 == NULL) {

printf("ERROR:map view\n");

return FALSE;

}

int distanceToStartRead = indexRead2 - startMapping2;

readingInMap2 = (TCHAR\*)pointerMap2 + distanceToStartRead;

int sizeReadingInMap = sizeView2 - distanceToStartRead;

for (int i = 0; i < sizeReadingInMap; i++) {

if (indexInRepl < sizeRepl)

readingInMap2[i] = repl[indexInRepl];

else {

notEnd = FALSE;

break;

}

indexInRepl++;

}

/\*if ((startMapping2 + sizeView2) == mapRepl->sizeFile)

notEnd = FALSE;\*/

if (!UnmapViewOfFile(pointerMap2)) {

printf("ERROR: unmap view\n");

return FALSE;

}

indexRead2 += sizeReadingInMap;

}

if ((startMapping2 + sizeRepl) == mapRepl->sizeFile)

notEnd = FALSE;

else

notEnd = TRUE;

indexRead2 = startRepl + sizeRepl;

indexRead1 = startRepl + sizeSample;

while (notEnd) {

startMapping1 = (indexRead1 / sysGranularity) \* sysGranularity;

sizeView1 = (indexRead1 % sysGranularity) + ((indexRead1 + map->sizeMemory) < map->sizeFile ? map->sizeMemory : (map->sizeFile - indexRead1));

pointerMap1 = MapViewOfFile(map->hFileMap, FILE\_MAP\_READ, (DWORD)(indexRead1 >> 0x20), startMapping1, sizeView1);

if (pointerMap1 == NULL) {

printf("ERROR:map view\n");

return FALSE;

}

int distanceToStartRead1 = indexRead1 - startMapping1;

readingInMap1 = (TCHAR\*)pointerMap1 + distanceToStartRead1;

int sizeReadingInMap = sizeView1 - distanceToStartRead1;

startMapping2 = (indexRead2 / sysGranularity) \* sysGranularity;

sizeView2 = (indexRead2 % sysGranularity) + ((indexRead2 + mapRepl->sizeMemory) < mapRepl->sizeFile ? mapRepl->sizeMemory : (mapRepl->sizeFile - indexRead2));

pointerMap2 = MapViewOfFile(mapRepl->hFileMap, FILE\_MAP\_WRITE, (DWORD)(indexRead2 >> 0x20), startMapping2, sizeView2);

if (pointerMap2 == NULL) {

printf("ERROR:map view\n");

return FALSE;

}

int distanceToStartRead2 = indexRead2 - startMapping2;

readingInMap2 = (TCHAR\*)pointerMap2 + distanceToStartRead2;

for (int i = 0; i < sizeReadingInMap; i++)

readingInMap2[i] = readingInMap1[i];

if ((startMapping2 + sizeView2) == mapRepl->sizeFile)

notEnd = FALSE;

if (!UnmapViewOfFile(pointerMap1)) {

printf("ERROR: unmap view\n");

return FALSE;

}

if (!UnmapViewOfFile(pointerMap2)) {

printf("ERROR: unmap view\n");

return FALSE;

}

indexRead1 += sizeReadingInMap;

indexRead2 += sizeReadingInMap;

}

if (!CloseFileMap(map))

return FALSE;

if (!CloseFileMap(mapRepl))

return FALSE;

DeleteFile(map->nameFile);

MoveFile(mapRepl->nameFile, map->nameFile);

map->hFile = CreateFile(map->nameFile, GENERIC\_READ, 0, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL);

if (map->hFile == INVALID\_HANDLE\_VALUE) {

printf("ERROR: can't open file\n");

return FALSE;

}

map->hFileMap = CreateFileMapping(map->hFile, NULL, PAGE\_READONLY, 0, 0, NULL);

if (map->hFileMap == NULL) {

printf("ERROR: create mapping\n");

return FALSE;

}

map->sizeFile = mapRepl->sizeFile;

free(mapRepl);

return TRUE;

}

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <windows.h>

#include <stdio.h>

#include <tchar.h>

#include <string.h>

#include "mapping.h"

int i;

TCHAR\* sample = 0;

TCHAR\* repl = 0;

DWORD sizeSample = 0;

DWORD sizeRepl = 0;

DWORD64 numLine = 0;

BOOL reg = TRUE;

TCHAR str[50],strd[50];

BOOL Cmpeq(const TCHAR\* str1, const TCHAR\* str2) {

int n = strlen(str2);

if (!strncmp((char\*)str1, (char\*)str2,n))

return TRUE;

else

return FALSE;

}

DWORD ParseCom(const TCHAR \*argv[]) {

const TCHAR\* com = argv[0];

if (Cmpeq(com, TEXT("/com:getLine"))) {

sscanf((char\*)argv[1], "%lu", &numLine);

return 1;

}

else if (Cmpeq(com, TEXT("/com:searchR")) || Cmpeq(com, TEXT("/com:search"))) {

if (Cmpeq(com, TEXT("/com:searchR")))

reg = FALSE;

else

reg = TRUE;

sample = (TCHAR\*)malloc(sizeof(TCHAR));

for (sizeSample = 0; argv[1][sizeSample] != '\000'; (sizeSample)++) {

sample = (TCHAR\*)realloc(sample, sizeof(TCHAR)\* (sizeSample + 1));

sample[sizeSample] = argv[1][sizeSample];

}

return 2;

}

else if (Cmpeq(com, TEXT("/com:replace"))) {

sample = (TCHAR\*)malloc(sizeof(TCHAR));

for (sizeSample = 0; argv[1][sizeSample] != '\000'; (sizeSample)++) {

sample = (TCHAR\*)realloc(sample, sizeof(TCHAR)\* (sizeSample + 1));

sample[sizeSample] = argv[1][sizeSample];

}

repl = (TCHAR\*)malloc(sizeof(TCHAR));

for (sizeRepl = 0; argv[2][sizeRepl] != '\000'; (sizeRepl)++) {

repl = (TCHAR\*)realloc(repl, sizeof(TCHAR)\* (sizeRepl + 1));

repl[sizeRepl] = argv[2][sizeRepl];

}

return 3;

}

return 0;

}

void help() {

printf("\t one command mode\n");

printf("/f:[name file] - select file\n");

printf("/com:getLine [number Line] - get line\n");

printf("/com:search [\"sample\"] - search\n");

printf("/com:searchR [\"sample\"] - search not register\n");

printf("/com:replace [\"sample\"] [\"replace\"] - replacing\n");

printf("/min:[number] - min size file\n");

printf("/max:[number] - max size file\n");

printf("/ram:[number] - max ram (not less Granularity)\n");

printf("/info - information file and program settings\n");

printf("/inter - interactive mode (turns on if there is no /com)\n");

printf("/MD5 - check sum\n");

printf("/help - help\n");

printf("\n\t interactive mode\n");

printf("/f:[name file] - select file\n");

printf("/getLine [number Line] - get line\n");

printf("/search [\"sample\"] - search\n");

printf("/searchR [\"sample\"] - search not register\n");

printf("/replace [\"sample\"] [\"replace\"] - replacing\n");

printf("/min:[number] - min size file\n");

printf("/max:[number] - max size file\n");

printf("/ram:[number] - max ram (not less Granularity)\n");

printf("/info - information file and program settings\n");

printf("/inter - interactive mode (turns on if there is no /com)\n");

printf("/MD5 - check sum\n");

printf("/help - help\n");

printf("/exit\n");

}

int \_tmain(int argc, TCHAR \*argv[]) {

TCHAR\* fileName = NULL;

SYSTEM\_INFO sysInfo;

GetSystemInfo(&sysInfo);

DWORD sysGranularity = sysInfo.dwAllocationGranularity;

DWORD64 minSizeFile = 0x1;

DWORD64 maxSizeFile = 0xFFFFFFFFFFFFFFFF;

DWORD maxSizeRam = 262144;

BOOL bInteractive = TRUE;

BOOL info = FALSE;

DWORD numCommand = 0;

//DWORD64 numLine = 0;

//DWORD sizeSample;

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

if (Cmpeq(argv[i],TEXT("/com"))) {

if (argv[i][4] != ':' || !argv[i][5]) {

printf("ERROR: invalid usage of /com\n");

return 1;

}

numCommand = ParseCom(&argv[i]);

if (!numCommand) {

printf("ERROR: invalid com\n");

return 1;

}

}

else if (Cmpeq(argv[i], TEXT("/f"))) {

if (argv[i][2] != ':' || !argv[i][2]) {

printf("ERROR: invalid usage of /f\n");

return 2;

}

fileName = argv[i] + 3;

}

else if (Cmpeq(argv[i], TEXT("/help"))) {

help();

//return 0;

}

else if (Cmpeq(argv[i], TEXT("/min"))) {

if (argv[i][4] != ':' || !argv[i][5]) {

printf("ERROR: invalid usage of /min\n");

return 3;

}

sscanf((char\*)argv[i] + 5, "%llu", &minSizeFile);

if (!minSizeFile) {

printf("ERROR: invalid size specified\n");

return 3;

}

}

else if (Cmpeq(argv[i], TEXT("/max"))) {

if (argv[i][4] != ':' || !argv[i][5]) {

printf("ERROR: invalid usage of /min\n");

return 4;

}

sscanf((char\*)argv[i] + 5, "%llu", &maxSizeFile);

if (!maxSizeFile) {

printf("ERROR: invalid size specified\n");

return 4;

}

}

else if (Cmpeq(argv[i], TEXT("/ram"))) {

if (argv[i][4] != ':' || !argv[i][5]) {

printf("ERROR: invalid usage of /ram\n");

return 5;

}

sscanf((char\*)argv[i] + 5, "%lu", &maxSizeRam);

if (!maxSizeRam) {

printf("ERROR: invalid size specified\n");

return 5;

}

if (maxSizeRam < sysGranularity)

maxSizeRam = sysGranularity;

}

else if (Cmpeq(argv[i], TEXT("/inter"))) {

bInteractive = TRUE;

}

else if (Cmpeq(argv[i], TEXT("/info"))) {

info = TRUE;

}

}

bInteractive = bInteractive && !numCommand;

if (bInteractive) {

TCHAR cmq[56];

DWORD64 i;

DWORD ram;

TMapping\* map = malloc(sizeof(TMapping));

map->hFile = NULL;

map->hFileMap = NULL;

BOOL exit = TRUE;

if (fileName) {

if (!CreateFileMap(map, fileName, maxSizeRam, minSizeFile, maxSizeFile))

return -1;

}

while (exit) {

numCommand = 0;

scanf("%s", cmq);

if (Cmpeq(cmq, TEXT("/f:"))) {

int n = strlen(cmq);

fileName = malloc(sizeof(char)\* (n - 2));

sscanf((char\*)cmq + 3, "%s", fileName);

sprintf(str,"CertUtil -hashfile %s MD5", fileName);

system(str);

numCommand = 4;

}

else if (Cmpeq(cmq, TEXT("/getLine"))) {

scanf("%s", cmq);

sscanf((char\*)cmq, "%lu", &numLine);

numCommand = 1;

}

else if (Cmpeq(cmq, TEXT("/searchR")) || Cmpeq(cmq, TEXT("/search"))) {

if (Cmpeq(cmq, TEXT("/searchR")))

reg = FALSE;

else

reg = TRUE;

sample = (TCHAR\*)malloc(sizeof(TCHAR));

getchar();

TCHAR c = getchar();

if (c != '"') {

printf("ERROR: invalid usage of /com:search\n");

numCommand = 0;

continue;

}

for (sizeSample = 0; (c = getchar()) != '"'; (sizeSample)++) {

sample = (TCHAR\*)realloc(sample, sizeof(TCHAR)\* (sizeSample + 1));

sample[sizeSample] = c;

}

numCommand = 2;

}

else if (Cmpeq(cmq, TEXT("/replace"))) {

sample = (TCHAR\*)malloc(sizeof(TCHAR));

getchar();

TCHAR c = getchar();

if (c != '"') {

printf("ERROR: invalid usage of /com:replace\n");

numCommand = 0;

continue;

}

for (sizeSample = 0; (c = getchar()) != '"'; (sizeSample)++) {

sample = (TCHAR\*)realloc(sample, sizeof(TCHAR)\* (sizeSample + 1));

sample[sizeSample] = c;

}

getchar();

c = getchar();

if (c != '"') {

printf("ERROR: invalid usage of /com:replace\n");

numCommand = 0;

continue;

}

for (sizeRepl = 0; (c = getchar()) != '"'; (sizeRepl)++) {

repl = (TCHAR\*)realloc(repl, sizeof(TCHAR)\* (sizeRepl + 1));

repl[sizeRepl] = c;

}

numCommand = 3;

}

else if (Cmpeq(cmq, TEXT("/help"))) {

help();

numCommand = 0;

}

else if (Cmpeq(cmq, TEXT("/min"))) {

if (cmq[4] != ':' || !cmq[5]) {

printf("ERROR: invalid usage of /min\n");

numCommand = 0;

}

sscanf((char\*)cmq + 5, "%llu", &minSizeFile);

if (!minSizeFile) {

printf("ERROR: invalid size specified\n");

numCommand = 0;

}

}

else if (Cmpeq(cmq, TEXT("/max"))) {

if (cmq[4] != ':' || !cmq[5]) {

printf("ERROR: invalid usage of /min\n");

numCommand = 0;

}

sscanf((char\*)cmq + 5, "%llu", &maxSizeFile);

if (!maxSizeFile) {

printf("ERROR: invalid size specified\n");

numCommand = 0;

}

}

else if (Cmpeq(cmq, TEXT("/ram"))) {

if (cmq[4] != ':' || !cmq[5]) {

printf("ERROR: invalid usage of /ram\n");

numCommand = 0;

}

sscanf((char\*)cmq + 5, "%lu", &maxSizeRam);

if (!maxSizeRam) {

printf("ERROR: invalid size specified\n");

numCommand = 0;

}

if (maxSizeRam < sysGranularity)

maxSizeRam = sysGranularity;

}

else if (Cmpeq(cmq, TEXT("/exit"))) {

numCommand = 5;

}

else if (Cmpeq(cmq, TEXT("/info"))) {

numCommand = 6;

}

else if (Cmpeq(cmq, TEXT("/MD5"))) {

numCommand = 7;

}

switch (numCommand) {

case 4:

if (map->hFileMap)

if (!CloseFileMap(map))

return -2;

if (!CreateFileMap(map, fileName, maxSizeRam, minSizeFile, maxSizeFile))

continue;

else

printf("ok\n");

break;

case 1:

if (!map->hFile) {

printf("file is not selected\n");

continue;

}

if (!GetLine(map, numLine, sysGranularity))

continue;

printf("\n");

break;

case 2:

if (!map->hFile) {

printf("file is not selected\n");

continue;

}

i = Search(map, sizeSample, sample,reg, sysGranularity);

if (i == -1)

continue;

if (i == -2)

printf("sample not search\n");

else

printf("%llu\n", i);

break;

case 3:

if (!map->hFile) {

printf("file is not selected\n");

continue;

}

ram = maxSizeRam;

if ((maxSizeRam / 2) < sysGranularity)

maxSizeRam = 2 \* sysGranularity;

if (!CloseFileMap(map))

return -2;

if (!CreateFileMap(map, fileName, maxSizeRam / 2, minSizeFile, maxSizeFile))

continue;

if (!Replace(map, sizeSample, sample, sizeRepl, repl, sysGranularity, minSizeFile, maxSizeFile))

continue;

else

printf("ok\n");

maxSizeRam = ram;

break;

case 5:

if (map->hFileMap)

if (!CloseFileMap(map))

return -3;

exit = FALSE;

break;

case 6:

if (!map->hFile) {

printf("file is not selected\n");

continue;

}

i = FileNumLine(map, sysGranularity);

if (i == 0)

return -7;

printf("name file %s\n", map->nameFile);

printf("size file %llu\n", map->sizeFile);

printf("quantity line %llu\n", i);

printf("max size file %llu\n", maxSizeFile);

printf("min size file %llu\n", minSizeFile);

printf("max ram %lu\n", maxSizeRam);

printf("\n");

break;

case 7:

sprintf(strd, "CertUtil - hashfile %s MD5", fileName);

if (strd != str)

{

printf("file was modifyed\n");

}

else printf("ok\n");

break;

}}

}

else {

// printf("min size f %llu\n max size f %llu\n max ROM %llu\n", minSizeFile, maxSizeFile, maxSizeRam);

if (!fileName) {

printf("file is not selected\n");

return 0;

}

DWORD64 i;

DWORD ram;

TMapping\* map = malloc(sizeof(TMapping));

map->hFile = NULL;

map->hFileMap = NULL;

switch (numCommand) {

case 1:

if (!CreateFileMap(map, fileName, maxSizeRam, minSizeFile, maxSizeFile))

return -1;

if (!GetLine(map, numLine, sysGranularity))

return -11;

printf("\n");

if (info) {

i = FileNumLine(map, sysGranularity);

if (i == 0)

return -7;

printf("name file %s\n", map->nameFile);

printf("size file %llu\n", map->sizeFile);

printf("quantity line %llu\n", i);

printf("max size file %llu\n", maxSizeFile);

printf("min size file %llu\n", minSizeFile);

printf("max ram %lu\n", maxSizeRam);

printf("\n");

break;

}

if (!CloseFileMap(map))

return -3;

break;

case 2:

if (!CreateFileMap(map, fileName, maxSizeRam, minSizeFile, maxSizeFile))

return -1;

i = Search(map, sizeSample, sample, reg, sysGranularity);

if (i == -1)

return -21;

if (i == -2)

printf("sample not search\n");

else

printf("%llu\n", i);

if (info) {

i = FileNumLine(map, sysGranularity);

if (i == 0)

return -7;

printf("name file %s\n", map->nameFile);

printf("size file %llu\n", map->sizeFile);

printf("quantity line %llu\n", i);

printf("max size file %llu\n", maxSizeFile);

printf("min size file %llu\n", minSizeFile);

printf("max ram %lu\n", maxSizeRam);

printf("\n");

break;

}

if (!CloseFileMap(map))

return -3;

break;

case 3:

ram = maxSizeRam;

if ((maxSizeRam / 2) < sysGranularity)

maxSizeRam = 2 \* sysGranularity;

if (map->hFileMap)

if (!CloseFileMap(map))

return -2;

if (!CreateFileMap(map, fileName, maxSizeRam / 2, minSizeFile, maxSizeFile))

return -1;

if (!Replace(map,sizeSample,sample,sizeRepl,repl,sysGranularity,minSizeFile,maxSizeFile))

return -31;

maxSizeRam = ram;

if (info) {

i = FileNumLine(map, sysGranularity);

if (i == 0)

return -7;

printf("name file %s\n", map->nameFile);

printf("size file %llu\n", map->sizeFile);

printf("quantity line %llu\n", i);

printf("max size file %llu\n", maxSizeFile);

printf("min size file %llu\n", minSizeFile);

printf("max ram %lu\n", maxSizeRam);

printf("\n");

break;

}

if (!CloseFileMap(map))

return -3;

break;

case 0:

if (info) {

i = FileNumLine(map, sysGranularity);

if (i == 0)

return -7;

printf("name file %s\n", map->nameFile);

printf("size file %llu\n", map->sizeFile);

printf("quantity line %llu\n", i);

printf("max size file %llu\n", maxSizeFile);

printf("min size file %llu\n", minSizeFile);

printf("max ram %lu\n", maxSizeRam);

printf("\n");

break;

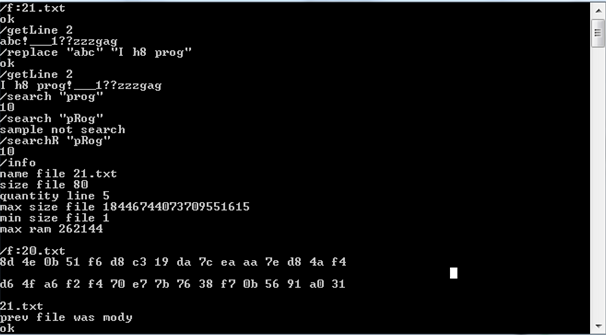
}

}

}

return 0;

}





Вывод

Проверка файла на изменение происходит автоматически при чтении следующего файла.

В конечном итоге получаем программу обработки текстовых файлов, которую можно запустить на ранних версиях Windows за счет маппинга файла, поэтому объем оперативной памяти не является решающим фактором