

Московский Авиационный Институт
(Национальный Исследовательский Университет)
Институт № 8 информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работ №3 по курсу
«Операционные системы»

Управление процессами в ОС

Студент: Пермяков Никита Александрович
Группа: М80 – 208Б-19
Вариант: 3
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2020

Содержание

1. Постановка задачи
2. Общие сведения о программе
3. Общий метод и алгоритм решения
4. Основные файлы программы
5. Демонстрация работы программы
6. Вывод

Постановка задачи.

Написать программу на языке Си по работе с очередью. Команды считываются в родительском процессе и поступают на обработку в дочерний процесс, в котором происходят все вычисления, после чего ответ поступает в родительский процесс. В результате работы программа (основной процесс) должен создать для решение задачи один или несколько дочерних процессов. Взаимодействие между процессами осуществляется через системные сигналы/события и/или каналы (pipe).

Необходимо обрабатывать системные ошибки, которые могут возникнуть в результате работы.

Общие сведения о программе

Программа компилируется из двух файлов: parent.cpp и child.cpp. Не используются заголовочные файлы. В программе используются следующие системные вызовы:

1. **read** – для чтения данных из файла.
2. **write** – для записи данных в файл.
3. **pipe** – для создания однонаправленного канала, через который могут общаться два процесса. Возвращает два дескриптора файлов: для чтения и для записи.
4. **fork** – для создания дочернего процесса.
5. **close** – для закрытия файла после окончания работы с ним.

Общий метод и алгоритм решения.

Пользователь вводит команды вида: «число число число <endline>». Далее эти числа передаются от родительского процесса в дочерний. Дочерний процесс производит деление первого числа, на последующие, а результат выводит в файл. Если происходит деление на 0, то тогда дочерний и родительский процесс завершают свою работу. Проверка деления на 0 должна осуществляться на стороне дочернего процесса. Числа имеют тип int.

Файлы программы.

Файл parent.cpp

```
#include
"stdio.h"

#include "windows.h"
#include <conio.h>
#include <tchar.h>

HANDLE hEventChar;
HANDLE hEventTermination;
HANDLE hEvents[2];

CHAR lpEventName[] = "nameSync";
CHAR lpEventTerminationName[] = "nameEnd";
CHAR lpFileShareName[] = "nameFile";

HANDLE hFileMapping;
```

```
LPVOID lpFileMap;
```

```
int main() {
```

```
    CHAR chr;
```

```
    int count_request = 0;
```

```
    int num = 0;
```

```
    int ch_end = 0;
```

```
    int x, result;
```

```
    DWORD dwRetCode;
```

```
    hEventChar = CreateEvent(NULL, FALSE, FALSE, lpEventName);
```

```
    if (hEventChar == NULL) {
```

```
        fprintf(stdout, "CreateEvent: Error %ld\n", GetLastError());
```

```
        _getch();
```

```
        return 0;
```

```
    }
```

```
    if (GetLastError() == ERROR_ALREADY_EXISTS) {
```

```
        printf("\nApplication EVENT already started\n"
```

```
            "Press any key to exit...");
```

```
        _getch();
```

```
        return 0;
```

```
    }
```

```
    hEventTermination = CreateEvent(NULL, FALSE, FALSE,  
lpEventTerminationName);
```

```
    if (hEventTermination == NULL) {  
        fprintf(stdout, "CreateEvent (Termination): Error %ld\n",  
GetLastError());  
        _getch();  
        return 0;  
    }
```

```
    hFileMapping = CreateFileMapping(INVALID_HANDLE_VALUE, NULL,  
PAGE_READWRITE, 0, 256, lpFileShareName);
```

```
    if (hFileMapping == NULL) {  
        fprintf(stdout, "CreateFileMapping: Error %ld\n",  
GetLastError());  
        _getch();  
        return 0;  
    }
```

```
    lpFileMap = MapViewOfFile(hFileMapping, FILE_MAP_READ |  
FILE_MAP_WRITE, 0, 0, 0);
```

```
    if (lpFileMap == 0) {  
        fprintf(stdout, "MapViewOfFile: Error %ld\n",  
GetLastError());  
        _getch();  
        return 0;  
    }
```

```
    hEvents[0] = hEventTermination;
```

```
    hEvents[1] = hEventChar;
```

```
    while (TRUE) {
```

```

chr = _getche();

if (chr == 0x20 || chr == 0X0A) {
    x = num;

    *((LPSTR)lpFileMap) = x;

    dwRetCode = WaitForMultipleObjects(2, hEvents, FALSE,
INFINITE);

    if (dwRetCode == WAIT_ABANDONED_0 ||
        dwRetCode == WAIT_ABANDONED_0 + 1 ||
        dwRetCode == WAIT_OBJECT_0 ||
        dwRetCode == WAIT_FAILED)
        break;

    if (count_request != 0) {
        result = *((LPSTR)lpFileMap);
        if ((char)result != 'E') {
            printf("Result call %d: %d\n", count_request,
result);
        } else {
            printf("Error from child proccess\n");
        }
    }
    count_request++;

    if (chr == 0x20) {
        num = 0;
        ch_end = 0;
    }
}

```

```

        continue;
    }

    if (chr == 0x0A) {
        if (ch_end == 2)
            break;
        ++ch_end;
    }
}

num = 10 * num + (chr - '0');
}

printf("Finish calc\n");

CloseHandle(hEventChar);
CloseHandle(hEventTermination);
UnmapViewOfFile(lpFileMap);
CloseHandle(hFileMapping);

return 0;
}

```

Файл child.cpp

```

#include
<windows.h>

#include <stdio.h>
#include <conio.h>
#include <tchar.h>

```



```
HANDLE hEvent;
```

```
HANDLE hEventTermination;
```

```
CHAR lpEventName[] = "nameSync";
```

```
CHAR lpEventTerminationName[] = "nameEnd";
```

```
CHAR lpFileShareName[] = "nameFile";
```

```
HANDLE hFileMapping;
```

```
LPVOID lpFileMap;
```

```
int main() {
```

```
    CHAR chr;
```

```
    int x, divider = 0;
```

```
    int count_run = 0;
```

```
    int res;
```

```
    DWORD dwRetCode;
```

```
    hEvent = OpenEvent(EVENT_ALL_ACCESS, FALSE, lpEventName);
```

```
    if (hEvent == NULL) {
```

```
        fprintf(stdout, "OpenEvent: Error %ld\n", GetLastError());
```

```
        _getch();
```

```
        return 0;
```

```
    }
```

```

        hEventTermination = OpenEvent(EVENT_ALL_ACCESS, FALSE,
lpEventTerminationName);

        if (hEventTermination == NULL) {

            fprintf(stdout, "OpenEvent (Termination): Error %ld\n",
GetLastError());

            _getch();

            return 0;

        }

        hFileMapping = OpenFileMapping(FILE_MAP_READ | FILE_MAP_WRITE,
FALSE, lpFileShareName);

        if (hFileMapping == NULL) {

            fprintf(stdout, "OpenFileMapping: Error %ld\n",
GetLastError());

            _getch();

            return 0;

        }

        lpFileMap = MapViewOfFile(hFileMapping, FILE_MAP_READ |
FILE_MAP_WRITE, 0, 0, 0);

        if (lpFileMap == 0) {

            fprintf(stdout, "MapViewOfFile: Error %ld\n",
GetLastError());

            _getch();

            return 0;

        }

        while (TRUE) {

```

```

        x = *((LPSTR)lpFileMap);

        if (count_run == 0) {
            divider = x;
        }
        if (count_run > 0) {
            if (divider != 0) {
                res = x / divider;
                *((LPSTR)lpFileMap) = res;
            }
            else {
                *((LPSTR)lpFileMap) = 'E';
                //res = 0;
            }
        }
        count_run++;

        SetEvent(hEvent);
    }

    SetEvent(hEvent);
    SetEvent(hEventTermination);
    CloseHandle(hEvent);
    CloseHandle(hEventTermination);
    UnmapViewOfFile(lpFileMap);
    CloseHandle(hFileMapping);

    return 0;
}

```

Демонстрация работы программы.

3 9 76 343 875 2323 765 344

Result call 1: 3

Result call 2: 25

Result call 3: 114

Result call 4: 291

Result call 5: 774

Result call 6: 255

Result call 7: 114

Finish calc

45 6875 2348234 45 2 32 43 54 450 543

Result call 1: 152

Result call 2: 52182

Result call 3: 1

Result call 4: 0

Result call 5: 0

Result call 6: 0

Result call 7: 1

Result call 8: 10

Result call 9: 12

Finish calc

Создание дочернего процесса

The screenshot displays two windows from Sysinternals: Process Monitor and Process Tree.

Process Monitor (Left Window): Shows a list of events. The columns are Time, Process Name, PID, Operation, Path, Result, and Detail. The events are filtered to show only 'services.exe' processes. The operations include 'RegOpenKey', 'RegQueryValue', and 'RegCloseKey'. The results are mostly 'SUCCESS'. The detail column shows 'Desired Access: R...' and 'Type: REG_SZ, Le...'. The bottom status bar indicates 'Showing 63 081 of 1 269 092 events (4%)' and 'Backed by virtual memory'.

Process Tree (Right Window): Shows a tree view of running processes. The columns are Process, Description, Image Path, Life Time, Company, Owner, Command, Start Time, and End Time. The tree shows a hierarchy starting from 'parent.exe (22928)' which is a 'Console Window ...'. It has several child processes, including 'conhost.exe (918)', 'child.exe (9408)', 'wmiprvse.exe (10412)', 'smartscreen.exe (12592)', 'wmiprvse.exe (20032)', 'wmiprvse.exe (6928)', 'WUDFHost.exe (1240)', and 'svchost.exe (1344)'. The bottom status bar shows 'Go To Event', 'Include Process', and 'Include Subtree' buttons.

Frame,"Module","Location","Address","Path"

0,"FLTMGR.SYS","FltDecodeParameters + 0x1cfd","0xffffffff8072567fa8d","C:\Windows\System32\drivers\FLTMGR.SYS"

1,"FLTMGR.SYS","FltDecodeParameters + 0x1840","0xffffffff8072567f5d0","C:\Windows\System32\drivers\FLTMGR.SYS"

2,"FLTMGR.SYS","FltQueryInformationFile + 0x963","0xffffffff807256b7d13","C:\Windows\System32\drivers\FLTMGR.SYS"

3,"ntoskrnl.exe","IoCallDriver + 0x59","0xffffffff80727c37159","C:\Windows\system32\ntoskrnl.exe"

4,"ntoskrnl.exe","PsGetProcessWow64Process + 0x104","0xffffffff80727d0a404","C:\Windows\system32\ntoskrnl.exe"

5,"ntoskrnl.exe","SeQueryInformationToken + 0x22be","0xffffffff807282010be","C:\Windows\system32\ntoskrnl.exe"

6,"ntoskrnl.exe","SeUnlockSubjectContext + 0x85f","0xffffffff807281f666f","C:\Windows\system32\ntoskrnl.exe"

7,"ntoskrnl.exe","ObOpenObjectByNameEx + 0x201","0xffffffff807281f4ad1","C:\Windows\system32\ntoskrnl.exe"

8,"ntoskrnl.exe","Ordinal1 + 0x129b","0xffffffff807282b1afb","C:\Windows\system32\ntoskrnl.exe"

9,"ntoskrnl.exe","setjmpex + 0x7bb5","0xffffffff80727dd5355","C:\Windows\system32\ntoskrnl.exe"

10,"ntdll.dll","ZwQueryAttributesFile + 0x14","0x7ffe4ee5ce94","C:\Windows\System32\ntdll.dll"

11,"wow64.dll","Wow64ShallowThunkSIZE_T64TO32 + 0x330d","0x7ffe4cf163fd","C:\Windows\System32\wow64.dll"

12,"wow64.dll","Wow64SystemServiceEx + 0x153","0x7ffe4cf17123","C:\Windows\System32\wow64.dll"

13,"wow64cpu.dll","TurboDispatchJumpAddressEnd + 0xb","0x77d01783","C:\Windows\System32\wow64cpu.dll"

14,"wow64cpu.dll","BTCpuSimulate + 0x9","0x77d01199","C:\Windows\System32\wow64cpu.dll"

15,"wow64.dll","Wow64LdrpInitialize + 0x26a","0x7ffe4cf1c77a","C:\Windows\System32\wow64.dll"

16,"wow64.dll","Wow64LdrpInitialize + 0x127","0x7ffe4cf1c637","C:\Windows\System32\wow64.dll"

17,"ntdll.dll","LdrInitShimEngineDynamic + 0x3133","0x7ffe4ee93da3","C:\Windows\System32\ntdll.dll"

18,"ntdll.dll","memset + 0x1e5a1","0x7ffe4ee81ba1","C:\Windows\System32\ntdll.dll"

19,"ntdll.dll","LdrInitializeThunk + 0x63","0x7ffe4ee31e53","C:\Windows\System32\ntdll.dll"

20,"ntdll.dll","LdrInitializeThunk + 0xe","0x7ffe4ee31dfe","C:\Windows\System32\ntdll.dll"

21,"ntdll.dll","NtQueryAttributesFile + 0xc","0x77d832ec","C:\Windows\SysWOW64\ntdll.dll"

22,"ntdll.dll","RtlQueryUnbiasedInterruptTime + 0xd64","0x77d75644","C:\Windows\SysWOW64\ntdll.dll"

23,"ntdll.dll","RtlDosSearchPath_Ustr + 0x5d5","0x77d5f325","C:\Windows\SysWOW64\ntdll.dll"

24,"KernelBase.dll","SearchPathW + 0xf9","0x7700d719","C:\Windows\SysWOW64\KernelBase.dll"

25,"KernelBase.dll","CreateProcessInternalW + 0xc00","0x76ff95b0","C:\Windows\SysWOW64\KernelBase.dll"

26,"KernelBase.dll","CreateProcessW + 0x2c","0x76ff892c","C:\Windows\SysWOW64\KernelBase.dll"

27,"parent.exe","CreateChildProcess + 0xd3, C:\Users\permi\source\repos\university\os\2_child_parent_proc\parent\parent.cpp(99)","0x801853","C:\Users\permi\source\repos\university\os\2_child_parent_proc\Debug\parent.exe"

28,"parent.exe","main + 0xec, C:\Users\permi\source\repos\university\os\2_child_parent_proc\parent\parent.cpp(28)","0x801c8c","C:\Users\permi\source\repos\university\os\2_child_parent_proc\Debug\parent.exe"

29,"parent.exe","invoke_main + 0x33, d:\agent_work\3\s\src\vc\tools\crt\vcstartup\src\startup\exe_common.inl(77)","0x802793","C:\Users\permi\source\repos\university\os\2_child_parent_proc\Debug\parent.exe"

```
30,"parent.exe", "__scrt_common_main_seh + 0x157,  
d:\agent\_work\3\s\src\vctools\crt\vcstartup\src\startup\exe_common.inl(281)","0x8025e7","C:\Us-  
ers\permi\source\repos\university\os\2_child_parent_proc\Debug\parent.exe"  
31,"parent.exe", "__scrt_common_main + 0xd,  
d:\agent\_work\3\s\src\vctools\crt\vcstartup\src\startup\exe_common.inl(330)","0x80247d","C:\Us-  
ers\permi\source\repos\university\os\2_child_parent_proc\Debug\parent.exe"  
32,"parent.exe", "mainCRTStartup + 0x8,  
d:\agent\_work\3\s\src\vctools\crt\vcstartup\src\startup\exe_main.cpp(16)","0x802818","C:\Us-  
ers\permi\source\repos\university\os\2_child_parent_proc\Debug\parent.exe"  
33,"kernel32.dll", "BaseThreadInitThunk + 0x19", "0x77276359", "C:\Windows\SysWOW64\ker-  
nel32.dll"  
34,"ntdll.dll", "RtlGetAppContainerNamedObjectPath + 0xe4", "0x77d78944", "C:\Windows\Sys-  
WOW64\ntdll.dll"  
35,"ntdll.dll", "RtlGetAppContainerNamedObjectPath + 0xb4", "0x77d78914", "C:\Windows\Sys-  
WOW64\ntdll.dll"
```

Вывод

Программа позволяет считывать и обрабатывать команды от пользователя запуская дочерний процесс, для обработки вводимых данных, и возврата ответа.

Взаимодействие между процессами осуществляется через системные сигналы/события. В работе получены навыки программирования запросов — ответов, на подобии серверного взаимодействия между собой и клиентом.

В работе были усвоены отличия процессов разных ОС.

Microsoft Windows процесс, – это контейнер для потоков. Процесс-контейнер содержит как минимум один поток. Если потоков в процессе несколько, приложение (процесс) становится многопоточным.

В Linux каждый поток является процессом, и для того, чтобы создать новый поток, нужно создать новый процесс.

- В чем заключается преимущество многопоточности Linux перед многопроцессностью?

В многопоточных приложениях Linux для создания дополнительных потоков используются процессы, представляющие собой обычные дочерние процессы главного процесса, но они разделяют с главным процессом следующее:

- 1) адресное пространство
- 2) файловые дескрипторы
- 3) обработчики сигналов

Для обозначения процессов этого типа, применяется специальный термин – легкие процессы (lightweight processes). Этим процессам не нужно создавать собственную копию адресного пространства своего процесса - родителя, создание нового легкого процесса требует значительно меньших затрат, чем создание полновесного дочернего процесса.