

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

Лабораторная работ №2 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"

HANDLE g_hChildStd_IN_Rd = NULL;
HANDLE g_hChildStd_IN_Wr = NULL;
HANDLE g_hChildStd_OUT_Rd = NULL;
HANDLE g_hChildStd_OUT_Wr = NULL;

int CreateChildProcess();

int main()
{
    SECURITY_ATTRIBUTES saAttr;
    saAttr.nLength = sizeof(SECURITY_ATTRIBUTES);
    saAttr.bInheritHandle = TRUE;
    saAttr.lpSecurityDescriptor = NULL;

    if (!CreatePipe(&g_hChildStd_OUT_Rd, &g_hChildStd_OUT_Wr, &saAttr, 0))
        return -1;

    if (!SetHandleInformation(g_hChildStd_OUT_Rd, HANDLE_FLAG_INHERIT, 0))
        return -1;

    if (!CreatePipe(&g_hChildStd_IN_Rd, &g_hChildStd_IN_Wr, &saAttr, 0))
        return -1;
```

```

if (!SetHandleInformation(g_hChildStd_IN_Wr, HANDLE_FLAG_INHERIT, 0))
    return -1;

if (!CreateChildProcess())
    return -1;

DWORD dwWritten, dwRead;

char ch;
int num = 0;
int count_request = 0;

int x, result;

while (true) {
    scanf_s("%c", &ch, 1);

    if (ch == 0x20 || ch == 0x0A) { // code space key or code enter key
        x = num;

        WriteFile(g_hChildStd_IN_Wr, &x, sizeof(int), &dwWritten, NULL);

        if (count_request != 0) {
            if (!ReadFile(g_hChildStd_OUT_Rd, &result, sizeof(int), &dwRead, NULL))
                return -1;

            printf("Result call %d: %d\n", count_request, result);
        }
        count_request++;

        if (ch == 0x20) {
            num = 0;
            continue;
        }

        if (ch == 0x0A)
            break;
    }

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

```

```

    printf("Finish calc\n");

    return 0;
}

int CreateChildProcess()
{
    TCHAR szCmdline[] = TEXT("child.exe");
    PROCESS_INFORMATION piProcInfo;
    STARTUPINFO siStartInfo;
    BOOL bSuccess = FALSE;

    ZeroMemory(&piProcInfo, sizeof(PROCESS_INFORMATION));

    // Set up members of the STARTUPINFO structure.
    // This structure specifies the STDIN and STDOUT handles for redirection
    .

    ZeroMemory(&siStartInfo, sizeof(STARTUPINFO));
    siStartInfo.cb = sizeof(STARTUPINFO);
    siStartInfo.hStdError = g_hChildStd_OUT_Wr;
    siStartInfo.hStdOutput = g_hChildStd_OUT_Wr;
    siStartInfo.hStdInput = g_hChildStd_IN_Rd;
    siStartInfo.dwFlags |= STARTF_USESTDHANDLES;

    // Create the child process.

    bSuccess = CreateProcess(NULL,
        szCmdline,      // command line
        NULL,           // process security attributes
        NULL,           // primary thread security attributes
        TRUE,           // handles are inherited
        0,              // creation flags
        NULL,           // use parent's environment
        NULL,           // use parent's current directory
        &siStartInfo,    // STARTUPINFO pointer
        &piProcInfo);    // receives PROCESS_INFORMATION

    // If an error occurs, exit the application.
    if (!bSuccess)
        return -1;
    else
    {

```

```

        CloseHandle(piProcInfo.hProcess);
        CloseHandle(piProcInfo.hThread);

        CloseHandle(g_hChildStd_OUT_Wr);
        CloseHandle(g_hChildStd_IN_Rd);
    }
}

```

Файл child.cpp

```

#include "stdio.h"
#include "windows.h"

int main()
{
    HANDLE readHandle = GetStdHandle(STD_INPUT_HANDLE);
    HANDLE writeHandle = GetStdHandle(STD_OUTPUT_HANDLE);
    DWORD readedBytes, writedBytes;

    int x, divider = 0;
    int count_run = 0;

    int res;

    while (true) {
        if (!ReadFile(readHandle, &x, sizeof(int), &readedBytes, NULL))
            return -1;

        if (count_run == 0) {
            divider = x;
        }

        if (count_run > 0) {
            if (divider != 0)
                res = x / divider;
            else
                res = 0;
            WriteFile(writeHandle, &res, sizeof(int), &writedBytes, NULL);
        }
        count_run++;
    }

    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' on the path 'HKLM\System\CurrentControlSet\Services\'. The results are mostly 'SUCCESS', with some '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 'conhost' (Console Window) and 'code.exe' (Visual Studio Code). The 'parent.exe' (PID 22928) is highlighted. The 'Description' pane at the bottom shows details for the selected process: 'Company: C:\Users\pemi\source\repos\university\ps\2_child_parent_proc\Debug\parent.exe', 'Command: "C:\Users\pemi\source\repos\university\ps\2_child_parent_proc\Debug\parent.exe"', 'User: DELL\pemi', 'PID: 22928', and 'Started: 07.11.2020 18:17:42'.

Лог создания дочернего процесса

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\vctools\crt\vcstartup\src\startup\exe_common.inl(77)","0x802793","C:\Us-  
ers\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.