Assignement -4

1. **Basic Process Creation**

**Objective :**

**● Understanding process creation using fork() in C.**

**Task :**

**● Write a C program that creates a child process using fork().**

**● Print the process IDs (PID) of both parent and child.**

**● Use getpid() and getppid() to display process details.**

**Expected Output :**

**● Display parent and child PIDs.**

**● Identify whether execution happens in the child or parent process.**

#include <stdio.h>

#include <windows.h>

int main()

{

    STARTUPINFO si;

    PROCESS\_INFORMATION pi;

    ZeroMemory(&si, sizeof(si));

    si.cb = sizeof(si);

    ZeroMemory(&pi, sizeof(pi));

    if (!CreateProcess(

            NULL,

            "cmd /c echo Child Process Running",

            NULL,

            NULL,

            FALSE,

            0,

            NULL,

            NULL,

            &si,

            &pi

            )){

        printf("Process creation failed. Error: %d\n", GetLastError());

        return 1;

    }

    printf("Parent Process:\n");

    printf("PID: %lu\n", GetCurrentProcessId());

    printf("Child Process Created (PID: %lu)\n", pi.dwProcessId);

    WaitForSingleObject(pi.hProcess, INFINITE);

    CloseHandle(pi.hProcess);

    CloseHandle(pi.hThread);

    return 0;

}

**Output :**

=thread-group-added,id="i1"

GNU gdb (GDB) 7.6.1

Copyright (C) 2013 Free Software Foundation, Inc.

License GPLv3+: GNU GPL version 3 or later <http:*//gnu.org/licenses/gpl.html>*

This is free software: you are free to change and redistribute it.

There is NO WARRANTY, to the extent permitted by law.  Type "show copying"

and "show warranty" for details.

This GDB was configured as "mingw32".

For bug reporting instructions, please see:

<http:*//www.gnu.org/software/gdb/bugs/>.*

Warning: Debuggee TargetArchitecture not detected, assuming x86\_64.

=cmd-param-changed,param="pagination",value="off"

=cmd-param-changed,param="args",value="2>CON 1>CON <CON"

[New Thread 68308.0xcbec]

[New Thread 68308.0xddfc]

Loaded 'C:\Windows\SysWOW64\kernel32.dll'. Symbols loaded.

Loaded 'C:\Windows\SysWOW64\KernelBase.dll'. Symbols loaded.

Loaded 'C:\Windows\SysWOW64\apphelp.dll'. Symbols loaded.

Loaded 'C:\Windows\SysWOW64\msvcrt.dll'. Symbols loaded.

[New Thread 68308.0xa054]

Child Process Running

Parent Process:

PID: 68308

Child Process Created (PID: 62164)

The program 'D:\OScript\Assignment4\pr\_1.exe' has exited with code 0 (0x00000000).

[Running] cd "d:\OScript\Assignment4\" && gcc pr\_1.c -o pr\_1 && "d:\OScript\Assignment4\"pr\_1

Child Process Running

Parent Process:

PID: 24500

Child Process Created (PID: 49416)

[Done] exited with code=0 in 3.093 seconds

1. **Process Synchronization Using wait()**

**Objective :**

**● Understanding how a parent waits for a child process to finish execution.**

**Task :**

**● Modify the previous program to make the parent wait for the child to complete execution using wait() .**

**● Print messages before and after the child process exits.**

**Expected Output :**

**● The parent should wait for the child to finish before printing its final message.**

#include <stdio.h>

#include <windows.h>

int main()

{

    STARTUPINFO si;

    PROCESS\_INFORMATION pi;

    ZeroMemory(&si, sizeof(si));

    si.cb = sizeof(si);

    ZeroMemory(&pi, sizeof(pi));

    printf("Parent Process (PID: %lu) is creating a child process...\n", GetCurrentProcessId());

    if (!CreateProcess(

            NULL,

            "cmd /c echo Child Process Running",

            NULL,

            NULL,

            FALSE,

            0,

            NULL,

            NULL,

            &si,

            &pi

        )) {

        printf("Process creation failed. Error: %d\n", GetLastError());

        return 1;

    }

    printf("Child Process Created (PID: %lu)\n", pi.dwProcessId);

    printf("Parent Process is waiting for the child to complete...\n");

    WaitForSingleObject(pi.hProcess, INFINITE);

    printf("Child Process has finished execution.\n");

    printf("Parent Process (PID: %lu) resumes execution.\n", GetCurrentProcessId());

    CloseHandle(pi.hProcess);

    CloseHandle(pi.hThread);

    return 0;

}

**Output :**

PS D:\OScript>  & 'c:\Users\Khushi\.vscode\extensions\ms-vscode.cpptools-1.23.6-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-k35pw3y1.kcb' '--stdout=Microsoft-MIEngine-Out-w5kj5mew.nfu' '--stderr=Microsoft-MIEngine-Error-hf1qobia.1lw' '--pid=Microsoft-MIEngine-Pid-reqtjsqk.qp0' '--dbgExe=C:\MinGW\bin\gdb.exe' '--interpreter=mi'

[Running] cd "d:\OScript\Assignment4\" && gcc pr\_2.c -o pr\_2 && "d:\OScript\Assignment4\"pr\_2

Child Process Running

Parent Process (PID: 61520) is creating a child process...

Child Process Created (PID: 14316)

Parent Process is waiting for the child to complete...

Child Process has finished execution.

Parent Process (PID: 61520) resumes execution.

[Done] exited with code=0 in 3.23 seconds

1. **Process Execution using exec()**

**Objective :**

**● Executing a new program within a child process.**

**Task :**

**● Create a C program where the child process replaces itself with another program (e.g., /bin/ls) using execlp().**

**● The parent process waits for the child to complete execution.**

**Expected Output :**

**● The child process successfully replaces itself with the ls command output.**

#include <stdio.h>

#include <windows.h>

int main()

{

    STARTUPINFO si;

    PROCESS\_INFORMATION pi;

    ZeroMemory(&si, sizeof(si));

    si.cb = sizeof(si);

    ZeroMemory(&pi, sizeof(pi));

    printf("Parent Process (PID: %lu) is creating a child process...\n", GetCurrentProcessId());

    if (!CreateProcess(

            NULL,

            "cmd /c dir",

            NULL,

            NULL,

            FALSE,

            0,

            NULL,

            NULL,

            &si,

            &pi

        )) {

        printf("Process creation failed. Error: %d\n", GetLastError());

        return 1;

    }

    printf("Child Process Created (PID: %lu). Executing 'dir' command...\n", pi.dwProcessId);

    printf("Parent Process is waiting for the child to complete...\n");

    WaitForSingleObject(pi.hProcess, INFINITE);

    printf("Child Process has finished execution.\n");

    printf("Parent Process (PID: %lu) resumes execution.\n", GetCurrentProcessId());

    CloseHandle(pi.hProcess);

    CloseHandle(pi.hThread);

    return 0;

}

**Output :**

PS D:\OScript>  & 'c:\Users\Khushi\.vscode\extensions\ms-vscode.cpptools-1.23.6-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-o0y5y4bt.ibl' '--stdout=Microsoft-MIEngine-Out-xu2ztkto.5dd' '--stderr=Microsoft-MIEngine-Error-qeysp42x.mru' '--pid=Microsoft-MIEngine-Pid-sn4otmtn.kk2' '--dbgExe=C:\MinGW\bin\gdb.exe' '--interpreter=mi'

[Running] cd "d:\OScript\Assignment4\" && gcc pr\_3.c -o pr\_3 && "d:\OScript\Assignment4\"pr\_3

 Volume in drive D is New Volume

 Volume Serial Number is 3A17-096A

 Directory of d:\OScript\Assignment4

10-02-2025  13:36    <DIR>          .

10-02-2025  12:53    <DIR>          ..

10-02-2025  13:11             2,371 pr\_1.c

10-02-2025  13:09            50,768 pr\_1.exe

10-02-2025  13:18             1,598 pr\_2.c

10-02-2025  13:16            51,280 pr\_2.exe

10-02-2025  13:25             1,601 pr\_3.c

10-02-2025  13:36            42,944 pr\_3.exe

10-02-2025  13:35             1,056 pr\_child\_4.c

10-02-2025  13:33            42,128 pr\_child\_4.exe

10-02-2025  13:36             2,350 pr\_parent\_4.c

10-02-2025  13:32            42,562 pr\_parent\_4.exe

              10 File(s)        238,658 bytes

               2 Dir(s)  243,303,321,600 bytes free

Parent Process (PID: 71568) is creating a child process...

Child Process Created (PID: 28756). Executing 'dir' command...

Parent Process is waiting for the child to complete...

Child Process has finished execution.

Parent Process (PID: 71568) resumes execution.

[Done] exited with code=0 in 3.235 seconds

**4. Inter-Process Communication using Pipes**

**Objective :**

**● Using pipes for communication between processes.**

**Task :**

**● Create a parent and child process.**

**● The parent sends a message to the child process via a pipe.**

**● The child reads the message from the pipe and prints it.**

**Expected Output :**

**● The child successfully reads and displays the message sent by the parent.**

**Parent :**

#include <stdio.h>

#include <windows.h>

int main()

{

    HANDLE hRead, hWrite;

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

    char message*[]* = "Hello from Parent!";

    char buffer[128];

    DWORD bytesRead, bytesWritten;

    if (!CreatePipe(&hRead, &hWrite, &sa, 0))

{

        printf("Failed to create pipe. Error: %d\n", GetLastError());

        return 1;

    }

    STARTUPINFO si;

    PROCESS\_INFORMATION pi;

    ZeroMemory(&si, sizeof(si));

    si.cb = sizeof(si);

    ZeroMemory(&pi, sizeof(pi));

    si.hStdInput = hRead;

    si.dwFlags |= STARTF\_USESTDHANDLES;

    printf("Parent Process (PID: %lu) is creating a child process...\n", GetCurrentProcessId());

    if (!CreateProcess(

            NULL,

            "cmd /c child.exe",

            NULL,

            NULL,

            TRUE,

            0,

            NULL,

            NULL,

            &si,

            &pi

        )) {

        printf("Child process creation failed. Error: %d\n", GetLastError());

        return 1;

    }

    WriteFile(hWrite, message, sizeof(message), &bytesWritten, NULL);

    CloseHandle(hWrite);

    WaitForSingleObject(pi.hProcess, INFINITE);

    CloseHandle(pi.hProcess);

    CloseHandle(pi.hThread);

    CloseHandle(hRead);

    printf("Parent Process (PID: %lu) has finished.\n", GetCurrentProcessId());

    return 0;

}

**Output :**

PS D:\OScript>  & 'c:\Users\Khushi\.vscode\extensions\ms-vscode.cpptools-1.23.6-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-pok3d413.yib' '--stdout=Microsoft-MIEngine-Out-fmdmyw42.aqt' '--stderr=Microsoft-MIEngine-Error-a2mibnci.qwq' '--pid=Microsoft-MIEngine-Pid-0mdwynrg.hci' '--dbgExe=C:\MinGW\bin\gdb.exe' '--interpreter=mi'

Running] cd "d:\OScript\Assignment4\" && gcc pr\_parent\_4.c -o pr\_parent\_4 && "d:\OScript\Assignment4\"pr\_parent\_4

Parent Process (PID: 25488) is creating a child process...

Parent Process (PID: 25488) has finished.

[Done] exited with code=0 in 3.197 seconds

**Child :**

#include <stdio.h>

#include <windows.h>

int main() {

    char buffer[128];

    DWORD bytesRead;

    if (ReadFile(GetStdHandle(STD\_INPUT\_HANDLE), buffer, sizeof(buffer), &bytesRead, NULL))

{

        buffer[bytesRead] = '\0'; *// Null terminate the string*

        printf("Child Process (PID: %lu) received message: %s\n", GetCurrentProcessId(), buffer);

   }

else

{

        printf("Failed to read from pipe. Error: %d\n", GetLastError());

   }

    return 0;

}

**Output :**

PS D:\OScript>  & 'c:\Users\Khushi\.vscode\extensions\ms-vscode.cpptools-1.23.6-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-pok3d413.yib' '--stdout=Microsoft-MIEngine-Out-fmdmyw42.aqt' '--stderr=Microsoft-MIEngine-Error-a2mibnci.qwq' '--pid=Microsoft-MIEngine-Pid-0mdwynrg.hci' '--dbgExe=C:\MinGW\bin\gdb.exe' '--interpreter=mi'

Running] cd "d:\OScript\Assignment4\" && gcc pr\_parent\_4.c -o pr\_parent\_4 && "d:\OScript\Assignment4\"pr\_parent\_4

Parent Process (PID: 25488) is creating a child process...

Parent Process (PID: 25488) has finished.

[Done] exited with code=0 in 3.197 seconds

**5. Process Scheduling Simulation**

**Objective :**

**● Implementing a basic Round Robin scheduling algorithm.**

**Task :**

**● Simulate Round Robin scheduling for multiple processes with different burst times.**

**● Assume a time quantum and display process execution order.**

**Expected Output :**

**● A sequence of time slices showing process execution, demonstrating context switching.**

#include <stdio.h>

typedef struct {

    int id;

    int burstTime;

    int waitTime;

    int turnAround;

} Process;

void roundRobinScheduling(Process processes*[]*, int n, int quantum) {

    int time = 0, remaining = n;

    printf("\nProcess Execution Order:\n");

    while (remaining > 0) {

        for (int i = 0; i < n; i++) {

            if (processes[i].burstTime > 0) {

                printf("[ Time %d ] -> Process P%d executes\n", time, processes[i].id);

                int execTime = (processes[i].burstTime > quantum) ? quantum : processes[i].burstTime;

                time += execTime;

                processes[i].burstTime -= execTime;

                if (processes[i].burstTime == 0) {

                    remaining--;

                    processes[i].turnAround = time;

                    processes[i].waitTime = processes[i].turnAround - processes[i].burstTime;

                }

            }

        }

    }

    printf("\nProcess Completion Times:\n");

    printf("PID\tTurnaround Time\tWaiting Time\n");

    for (int i = 0; i < n; i++) {

        printf("P%d\t%d\t\t%d\n", processes[i].id, processes[i].turnAround, processes[i].waitTime);

    }

}

int main()

{

    int n, quantum;

    printf("Enter number of processes: ");

    scanf("%d", &n);

    Process processes[n];

    printf("Enter burst times for %d processes:\n", n);

    for (int i = 0; i < n; i++)

{

        processes[i].id = i + 1;

        printf("P%d: ", processes[i].id);

        scanf("%d", &processes[i].burstTime);

    }

    printf("Enter time quantum: ");

    scanf("%d", &quantum);

    roundRobinScheduling(processes, n, quantum);

    return 0;

}

**Output :**

PS D:\OScript>  & 'c:\Users\Khushi\.vscode\extensions\ms-vscode.cpptools-1.23.6-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-h2u3hxmo.1ah' '--stdout=Microsoft-MIEngine-Out-40mizlnq.q5b' '--stderr=Microsoft-MIEngine-Error-5jk4yok3.gax' '--pid=Microsoft-MIEngine-Pid-2qtg41z4.c5y' '--dbgExe=C:\MinGW\bin\gdb.exe' '--interpreter=mi'

[Running] cd "d:\OScript\Assignment4\" && gcc pr\_5.c -o pr\_5 && "d:\OScript\Assignment4\"pr\_5

Process Execution Order:

[ Time 0 ] -> Process P1 executes

[ Time 3 ] -> Process P2 executes

[ Time 6 ] -> Process P3 executes

[ Time 9 ] -> Process P1 executes

[ Time 12 ] -> Process P3 executes

[ Time 15 ] -> Process P1 executes

[ Time 18 ] -> Process P3 executes

Process Completion Times:

PID     Turnaround Time   Waiting Time

P1      15              7

P2      6                2

P3      18             9