Lab Assignment-12

ROLL: 2005535 | NAME: SAHIL SINGH | DATE: 12/04/22

QUES 1:

SOLUTION:

#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

*int* main()

{

    // make two process which run same

    // program after this instruction

    fork();

    printf("Hello world!\n");

    return 0;

}

OUTPUT:

Hello world!

Hello world!

QUES 2:

SOLUTION:

#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

*int* main()

{

    fork();

    fork();

    printf("hello\n");

    return 0;

}

OUTPUT:

hello

hello

hello

hello

QUES 3:

SOLUTION:

#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

*int* main()

{

    pid\_t p;

    p = fork();

    if (p == -1)

    {

        printf("There is an error while calling fork\n");

    }

    if (p == 0)

    {

        printf("We are in the child process\n");

    }

    else

    {

        printf("We are in the parent process\n");

    }

    return 0;

}

OUTPUT:

We are in the parent process

We are in the child process

QUES 4:

SOLUTION:

#include <unistd.h>

#include <stdio.h>

#include <sys/types.h>

*int* main()

{

*int* r;

    r = fork();

    if (r == 0)

    {

        printf("The process is child process\n");

        printf("child id = %d\n", getpid());

        printf("parent id = %d\n", getppid());

    }

    else

    {

        printf("The process is the parent process\n");

        printf("The process id = %d\n", getpid());

        printf("parent id = %d\n", getppid());

    }

    return 0;

}

OUTPUT:

The process is the parent process

The process id = 8414

parent id = 7043

The process is child process

child id = 8416

parent id = 2637

QUES 5:

SOLUTION:

#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

#define MAX\_COUNT 10

*void* ChildProcess(*void*);  /\* child process prototype \*/

*void* ParentProcess(*void*); /\* parent process prototype\*/

*int* main(*void*)

{

    pid\_t pid;

    pid = fork();

    if (pid == 0)

        ChildProcess();

    else

        ParentProcess();

    return 0;

}

*void* ChildProcess(*void*)

{

*int* i;

    for (i = 1; i <= MAX\_COUNT; i++)

        printf("This line is from child, value = %d\n", i);

    printf(" \*\*\* Child process is done \*\*\*\n");

}

*void* ParentProcess(*void*)

{

*int* i;

    for (i = 1; i <= MAX\_COUNT; i++)

        printf("This line is from parent, value = %d\n", i);

    printf("\*\*\* Parent is done \*\*\*\n");

}

OUTPUT:

This line is from parent, value = 1

This line is from parent, value = 2

This line is from parent, value = 3

This line is from parent, value = 4

This line is from parent, value = 5

This line is from parent, value = 6

This line is from parent, value = 7

This line is from parent, value = 8

This line is from parent, value = 9

This line is from parent, value = 10

\*\*\* Parent is done \*\*\*

This line is from child, value = 1

This line is from child, value = 2

This line is from child, value = 3

This line is from child, value = 4

This line is from child, value = 5

This line is from child, value = 6

This line is from child, value = 7

This line is from child, value = 8

This line is from child, value = 9

This line is from child, value = 10

 \*\*\* Child process is done \*\*\*

QUES 6:

SOLUTION:

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

*int* main()

{

*int* a;

*int* b;

*int* c;

*int* d;

*int* e;

*int* f;

*int* g;

*int* h;

*int* i;

    b = fork();

    if (b == 0) // it's child

    {

        d = fork();

        if (d == 0)

        {

            h = fork();

            if (h == 0)

            {

                i = fork();

                if (i == 0)

                    printf("%d: I\n", getpid());

                else

                    printf("%d: H\n", getpid());

            }

            else

                printf("%d: D\n", getpid());

        }

        else

        {

            e = fork();

            if (e == 0)

                printf("%d: E\n", getpid());

            else

            {

                f = fork();

                if (f == 0)

                    printf("%d: F\n", getpid());

                else

                    printf("%d: B\n", getpid());

            }

        }

    }

    else

    {

        c = fork();

        if (c == 0)

        {

            g = fork();

            if (g == 0)

                printf("%d: G\n", getpid());

            else

                printf("%d: C\n", getpid());

        }

        else

            printf("%d: A\n", getpid());

    }

    return 0;

}

OUTPUT:

9045: A

9047: C

9048: D

9046: B

9050: E

ss2402@heatcliff24:~/os\_lab/lab\_12$ 9051: H

9049: G

9053: I

9052: F

-----------------------------------------------------------------------------------------