Lab 4

1) clone didn't use any flags, therefore parent and child process ran in separate copies of memory space and writes are not visible to each other.

2) clone used CLONE\_VM and CLONE\_FILES flags. The parent and child process in same memory space. Memory writes are shared. File descriptors are shared by parent and child process. The file was closed by function do\_something.

**clone\_mpmProcess.c**

#define \_GNU\_SOURCE

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sched.h>

int x = 0, y = 0;

int do\_something()

{

    sleep(1);

printf ("\nThis is process(thread) %d. \n", getpid());

    x = 7;

    printf("x + y = %d\n", x + y);

    \_exit(0);

}

int main ( void )

{

    const int STACK\_SIZE = 65536;

    char \*stack, \*stackTop;

pid\_t pid, fpid;

    int cret;

    stack = malloc(STACK\_SIZE);

    stackTop = stack + STACK\_SIZE;

    cret = clone(do\_something,stackTop,0,NULL);

    if (cret == -1) {

        perror("clone");

        exit(0);

    }

printf ("\nThis is process (thread) %d.\n", getpid());

y = 1;

printf("x + y = %d\n", x + y);

}

**clone\_mpmThread.c**

#define \_GNU\_SOURCE

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sched.h>

int x = 0, y = 0;

int do\_something()

{

    sleep(1);

printf ("\nThis is process(thread) %d. \n", getpid());

    x = 7;

    printf("x + y = %d\n", x + y);

    \_exit(0);

}

int main ( void )

{

    const int STACK\_SIZE = 65536;

    char \*stack, \*stackTop;

pid\_t pid, fpid;

    int cret;

    stack = malloc(STACK\_SIZE);

    stackTop = stack + STACK\_SIZE;

    cret = clone(do\_something,stackTop,CLONE\_VM|CLONE\_FILES,NULL);

    if (cret == -1) {

        perror("clone");

        exit(0);

    }

printf ("\nThis is process (thread) %d.\n", getpid());

y = 1;

printf("x + y = %d\n", x + y);

}