**Student Name:** Jyotish Mandal

**Student ID:** 11617152

**Roll.no: A 12**

**Email Address:** [jotishmandal77177@gmail.com](mailto:jotishmandal77177@gmail.com)

**GitHub Link: https://github.com/JyotishM/OS-project**

**Code:**

**Process:**

#include <stdio.h>

#include <stdlib.h>

#include <limits.h>

#define MIN\_PID 100

#define MAX\_PID 1000

#define a CHAR\_BIT

int size = MAX\_PID - MIN\_PID + 1;

char \*b;

int allocate\_map();

int allocate\_pid();

void release\_pid(int pid);

int main()

{

int c = allocate\_map();

if (c == 1) {

printf("Bitmap Data Structure starts working\n\n");

int id = 0, i;

for(i=0; i<5;i++)

{

int d = allocate\_pid();

printf("Process %d: pid = %d\n", i+1, d);

}

release\_pid(100); printf("\nProcess having pid 100 is released.");

release\_pid(101); printf("\nProcess having pid 101 is released.");

release\_pid(102); printf("\nProcess having pid 102 is released.");

release\_pid(103); printf("\nProcess having pid 103 is released.");

release\_pid(104); printf("\nProcess having pid 104 is released.");

int d = allocate\_pid();

printf("\n\nProcess %d : pid = %d\n", ++i, d);

d = allocate\_pid();

printf("Process %d : pid = %d\n", ++i, d);

d = allocate\_pid();

printf("Process %d : pid = %d\n", ++i, d);

d = allocate\_pid();

printf("Process %d : pid = %d\n", ++i, d);

d = allocate\_pid();

printf("Process %d : pid = %d\n", ++i, d);

}

else printf("Bitmap Data Structure failed working\n");

}

int allocate\_map() {

b = (char\*)malloc((size+a-1)/a \* sizeof(char));

if (b) return 1;

return -1;

}

int allocate\_pid() {

int i = 0;

int pid = b[i/a] & (1 << (i & (a-1)));

while (pid != 0) {

i++;

pid = b[i/a] & (1 << (i & (a-1)));

}

if (i+MIN\_PID > MAX\_PID) return -1;

b[i/a] |= 1 << (i & (a-1));

return i+MIN\_PID;

}

void release\_pid(int pid) {

if (pid < 100) {

printf("\nInvalid PID: The range lies between 100 to 1000");

return;

}

int i = pid - MIN\_PID;

b[i/a] &= ~(1 << (i & (a-1)));

}

**Description:**

The process in Operating System is an instance of a computer program in execution and pid is a process identifier just to identify any active process. The above program is based on the PID manager i.e. it manages the process ids of the given processes. As we all know, all the processes created are assigned with a process id. The allocation and deallocation of the ids assigned with the processes implies to pid manager. And it keeps the track of the free pids as well.

Here, we need to use the data structure in order to organize the pids. Hence, according to Linux supportability, Bitmap type of data structure was used where value of i equal to 0 which means the pid of value i is available and 1 not available or used by some other process.

**Algorithm:**

**Description (purpose of use):**

**1)** #include<limits.h> : determines the values of various variables. Here, it is used to determine the char\_bit type.

**2)** int c = allocate\_map();

if (c == 1) {

printf("Bitmap Data Structure starts working\n\n");

int id = 0,

The bitmap type of data structure is initialized. It is used for memory organization. It is also known as bit array.

**3)** for(i=0; i<5;i++)

{

int d = allocate\_pid();

printf("Process %d: pid = %d\n", i+1, d);

}

Processes are created and allocates the pids.

**4)** release\_pid(100); printf("\nProcess having pid 100 is released.");

It is to release the pid of a process

**Description:**

The PIDs ranges from 100 to 1000 which is the process could be allocated with the pid from 100 to 1000 if it exceeds the range it will just return Invalid Pid, out of range.

Int allocate map() - creates and initialize a data structure for representing pids. It returns -1 if unsuccessful and 1 if successful.

Similarly, if i=0 indicates its free and 1 if unable to allocate pids

**Yes, I made the 5 revisions, which includes multiple approach to manage pid from which this project2.c file was found best.**

**Github Link: https://github.com/JyotishM/OS-project**