|  |  |
| --- | --- |
|  | #include<stdio.h> |
|  | #include<pthread.h> |
|  | #include<time.h>  #include<assert.h> |
|  |  |
|  | #define MIN\_PID 100 |
|  | #define MAX\_PID 1000 |
|  |  |
|  | pthread\_mutex\_t Lock; |
|  |  |
|  | struct process  { |
|  | int pid, assign; |
|  | }obj[MAX\_PID-MIN\_PID]; |
|  |  |
|  | int allocate\_map()  { |
|  | int i=MIN\_PID; |
|  | while(i<=MAX\_PID)  { |
|  | obj[i-MIN\_PID].pid=i; |
|  | obj[i-MIN\_PID].assign=0; |
|  | i++; |
|  | } |
|  | return 1; |
|  | } |
|  |  |
|  | int allocate\_pid()  { |
|  | int i=0; |
|  | while(i<MAX\_PID)  { |
|  | if (obj[i].assign==0)  { |
|  | pthread\_mutex\_lock(&Lock); |
|  | obj[i].assign=1; |
|  | pthread\_mutex\_unlock(&Lock); |
|  | return obj[i].pid; |
|  | } |
|  | i++; |
|  | } |
|  | return -1; |
|  | } |
|  |  |
|  | void release\_pid(int pid)  { |
|  | int i=0; |
|  | while(i<MAX\_PID)  { |
|  | if(obj[i]. pid==pid)  { |
|  | int res=pthread\_mutex\_lock(&Lock);  if(res==0)  {  printf(“Aborting the process as res==0”);  } |
|  | assert(res==0); |
|  | obj[i].assign=0; |
|  | res=pthread\_mutex\_unlock(&Lock); |
|  | break; |
|  | } |
|  | i++; |
|  | } |
|  | } |
|  |  |
|  | void\* Func(void \*args)  { |
|  | int pid=allocate\_pid(); |
|  | printf("New process created with pid: %d\n",pid); |
|  | int time=(rand()+20)%(40+1); |
|  | sleep(time); |
|  | release\_pid(pid); |
|  | printf("Process destroyed with pid: %d\n",pid); |
|  | } |
|  |  |
|  | void createProcess()  { |
|  | int i=0,n;  printf(“How many threads you want to create?”);  scanf(“%d”,&n); |
|  | pthread\_t threadPid[n]; |
|  | if(pthread\_mutex\_init(&Lock,NULL)!=0) |
|  | printf("Mutex init."); |
|  | while(i<n)  { |
|  | pthread\_create(&threadPid[i],NULL,Func,NULL); |
|  | sleep(1); |
|  | i++; |
|  | } |
|  |  |
|  | i=0; |
|  | while(i<n)  { |
|  | pthread\_join(threadPid[i],NULL); |
|  | i++; |
|  | } |
|  |  |
|  |  |

}

|  |
| --- |
| main() { |
|  | if(allocate\_map())  { |
|  | srand(time(0)); |
|  | createProcess(); |
|  | } |
|  | } |