프로세스 동기화(도전과제)

20185309

황명원

소스코드:

#include <pthread.h>  
#include <semaphore.h>  
#include <stdlib.h>  
#include <stdio.h>  
  
#define BufferSize 5 // Size of the buffer  
  
sem\_t semaphore;  
pthread\_mutex\_t mutex;  
  
int cnt;  
  
void \*func(void \*cnt\_num) {  
  
 for (int i = 0; i < 100000; i++) {  
 sem\_wait(&semaphore);  
 pthread\_mutex\_lock(&mutex);  
  
 cnt++;  
 printf("cnt%d :: %d\n", \*((int \*)cnt\_num),cnt);  
  
 pthread\_mutex\_unlock(&mutex);  
 sem\_post(&semaphore);  
  
 }  
}  
int main()  
{  
 pthread\_mutex\_init(&mutex, NULL);  
 sem\_init(&semaphore,0,BufferSize);  
  
 pthread\_t thread[5];  
 int iret[5]={1,2,3,4,5};  
  
 for(int i=0 ; i<5 ; i++){  
 pthread\_create(&thread[i], NULL, (void \*)func, (void \*)&iret[i]);  
 }  
 for(int i=0 ; i<5 ; i++){  
 pthread\_join(thread[i], NULL);  
 }  
  
 printf("Thread error: %d\n",500000-cnt);  
  
 exit(0);  
  
 pthread\_mutex\_destroy(&mutex);  
 sem\_destroy(&semaphore);  
  
 return 0;  
}

분석내용:

2개의 프로세스를 다룬다면 mutex 만 써도 되지만 2개이상의 프로세스를 다루는 상황으로는

세마포어를 사용하는것이 좋습니다.

모든 프로세스는 mutex라는 세마포어를 공유하며, mutex가 1로 초기화 됩니다.

wait 연산을 제일 먼저 실행하는 프로세스는 mutex 값이 1이므로 이 값을 0으로 줄이고 임계 구역 내부로 들어갈 수 있습니다. 이 프로세스가 임계 지역을 빠져나오면서 signal 연산을 통해서 mutex값을 0에서 1로 증가시키면 다른 프로세스가 임계 영역 내부로 들어올 수 있습니다.

세마포어를 통해 여러개 프로세스를 다루고 한개의 프로세스가 들어와 실행되는동안 다른 프로세스가 실행 못하게 합니다. 한개의 프로세스가 끝나면 다른 프로세스가 들어와서 실행하고 또 나머지 프로세스는 대기합니다. 이과정을 cnt가 500000이 될때까지 반복합니다.

실행결과:

텍스트이(가) 표시된 사진

자동 생성된 설명파일 생성

텍스트이(가) 표시된 사진

자동 생성된 설명실행(1)

(중략)

텍스트이(가) 표시된 사진

자동 생성된 설명실행(2)