운영체제

20135151 이갑성

1번

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <unistd.h>  #include <pthread.h>  #define NPROC 4  //쓰레드가 성공적으로 생성되었을때, 넘겨주는 쓰레드 식별번호를 저장하는 배열  pthread\_t threads[NPROC];  int done[NPROC];  // 쓰레드 함수  void \*thread\_main(void \*);  int main()  {  int i, rc, status;  printf("pid = %d \n", getpid()); //자신의 프로세스 pid를 출력한다.  //done배열은 0으로 저장해주고,  //thread\_main함수를 수행할 쓰레드를 만들어준다.  //쓰레드 함수를 실행시킬때 넘겨줄 인자값 i 넘겨준다.  //쓰레드가 성공적으로 생성 되면 쓰레드 식별번호를 threads 배열 i번째에 저장한다.  //i 와 i번째 쓰레드 식별번호를 출력한다.  for(i = 0; i < NPROC; i++){  done[i] = 0;  pthread\_create(&threads[i], NULL, &thread\_main, (void \*)(long)i);  printf("%d, %ld \n", i, threads[i]);  }  //done배열을 1으로 저장해주고,  //쓰레드 식별번호가 저장되있는 thread배열 3번째 부터 종료된 쓰레드를 정리해준다.  //만약 종료된 쓰레드를 정리하는 것을 실패 또는 성공하면 해당 문구로 알려준다.  for(i = (NPROC-1); i >= 0; i--){  done[i] = 1;  rc = pthread\_join(threads[i], (void \*\*)&status);  if(rc == 0){  printf("completed join with thread %d status: %d \n", i, status);  }  else{  printf("ERROR: return code from pthread\_join() is %d, thread %d \n", rc, i);  return -1;  }  }  return 0;  }  //쓰레드 함수  void \*thread\_main(void \*arg)  {  int i;  double result = 0.0;  //쓰레드를 생성시 인자값으로 받은 arg와 자신의 프로세스를 출력해준다.  printf("thread: %d, %d \n", \*((int \*)&arg), getpid());  //done배열에 arg(배열의 인덱스를 나타냄)번째의 값이 0일때  //연산을 수행하고 인자값과 결과값을 출력해준다.  while(!done[\*((int \*)&arg)]) {  for(i = 0; i < 1000000; i++){  result = result + (double)random();  }  printf("thread: %d, result: %e \n", \*((int \*)&arg),result);  }  //쓰레드 종료  pthread\_exit((void \*)0);  }  /\*  현재 동기화문제 때문에 실행 결과가 다르게 나오고 있는 상황이다.  쓰레드를 생성했을때 여러 쓰레드의 중첩 실행을 방지하기 위해서 lock을 이용하면  지금과 같은 현상을 방지할 수 있다.  \*/ |

2번

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <pthread.h>  #include <math.h>  #define L 128  typedef struct {  double volatile \*p\_s;  pthread\_mutex\_t \*p\_s\_lock;  int n;  int nproc;  double \*x;  double \*y;  int l;  } DATA;  //쓰레드 함수(인자값으로 A를 넘겨받는다)  void \*SMP\_scalprod(void \*arg)  {  register double localsum = 0.0;  long i;  DATA D = \*(DATA \*)arg; //인자값으로 받은 arg를 D에 대입해준다.  //D에 있는 값을 조건문으로 설정하고, for문을 통해 localsum을 계산 해준다.  for(i = D.n; i < D.l; i += D.nproc){  localsum += (D.x[i] + D.y[i]);  }  //쓰레드 충돌을 방지하기 위해 lock(문을 잠군다.)해준다.  //하나의 쓰레드가 사용중일때 다른 쓰레드는 들어오지 못한다.  pthread\_mutex\_lock(D.p\_s\_lock);  //for문을 통해 나온 결과 localsum을 최종 결과값 \*(D.p\_s)에 더해서 넣어준다.  \*(D.p\_s) += localsum;  //연산이 끝나면 다시 문을 열어준다.  //기다리고있는 다른 쓰레드가 들어올 수 있게 된다.  pthread\_mutex\_unlock(D.p\_s\_lock);  return NULL;  }  int main(int argc, char \*argv[]){  pthread\_t \*thread;  void \*retval;  int i, cpu;  DATA \*A;  volatile double s = 0.0;  pthread\_mutex\_t s\_lock;  double x[L], y[L];  //main함수에 전달된 인자의 개수가 2개가 아니라면 아래 해당문 출력후 종료한다.  if(argc != 2){  printf("usage: %s <number of CPU> \n", argv[0]);  exit(1);  }  cpu = atoi(argv[1]);//함수에 전달된 2번째 인자를 정수로 변환하여 cpu에 넣어준다.  thread = (pthread\_t \*)calloc(cpu, sizeof(pthread\_t));//pthread\_t 구조체를 cup 수만큼 동적 할>당해주고 주소값을 thread에 전달한다.  A = (DATA \*)calloc(cpu, sizeof(DATA));//DATA 구조체를 cup 수만큼 동적 할당해주고 주소값을 A에  전달해준다.  //배열 x y를 모두 1로 초기화 해주는 작업  for(i = 0; i < L; i++){  x[i] = y[i] = 1;  }  //잠금 객체 s\_lock을 만드는 함수(mutex 타입은 NULL로써 기본값 fast가 설정된다.)  pthread\_mutex\_init(&s\_lock, NULL);  //동적메모리 A(구조체 DATA 타입)가 현재 cpu개 만큼 방이 만들어져 있는 상태에서  //for문을 이용해서 각 메모리에 있는 값들을 넣어준다.  for(i = 0; i < cpu; i++){  A[i].n = i;  A[i].x = x;  A[i].y = y;  A[i].l = L;  A[i].nproc = cpu;  A[i].p\_s = &s;  A[i].p\_s\_lock = &s\_lock;  //SMP\_scalprod() 함수를 수행할 쓰레드를 만들어 주고 쓰레드 식별번호를  //동적 메모리 할당된 thread에 각각 넣어준다.  //위에서 데이터 값들을 넣었던 A는 인자값으로 넘겨준다.  if(pthread\_create(&thread[i], NULL, SMP\_scalprod, &A[i]))  {  perror("pthread\_create failed");  exit(1);  }  }  //for문을 통해 만들었던 쓰레드를 정리한다..  for(i = 0; i < cpu; i++)  {  if(pthread\_join(thread[i], &retval)){  perror("pthread\_join failed");  exit(1);  }  }  //연산했던 결과 s를 출력  printf("result of sum = %f \n", s);  exit(0);  }  //현재 코드는 멀티쓰레드를 구현했을때 충돌을 방지하기 위해 DATA라는 구조체를 이용하여  //동기화 문제를 해결해 주었다. 쓰레드를 생성할때 인자값으로 DATA 구조체 변수 A를 보냄으로써  //mytex 잠금과 해제를 연산할때 해주어 사용자가 원하는 연산계산 결과를 만들었다. |