스트 상자:      
     
스트 상자: 

12121518

유호균

OS 1차 과제

오퍼레이팅 시스템

-송민석 교수님

Multi – threaded sorting application

1. **프로그램 인터페이스**

main 함수의 argument로 입력파일의 이름을 입력받는다. 입력받은 파일을 읽어들여 멀티쓰레딩을 통해 배열을 정렬하고, 정렬된 각각의 배열을 merge하여 그 결과를 파일출력한다.

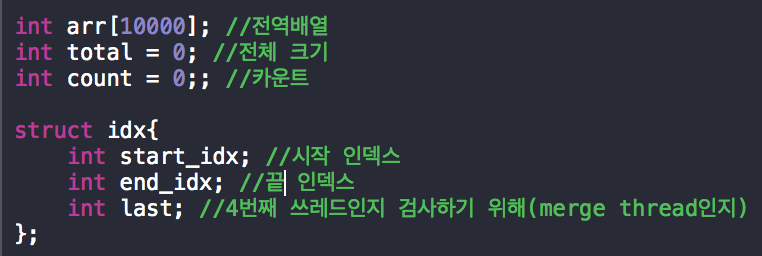


입력값은 main함수의 인자로 받기 때문에 어떠한 txt파일도 읽을 수 있다.

출력하는 파일은 output.txt로 정해져 있다.

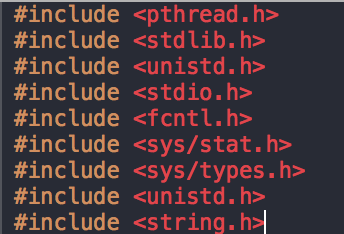
1. **프로그램 설명**

<전역 변수 및 struct>



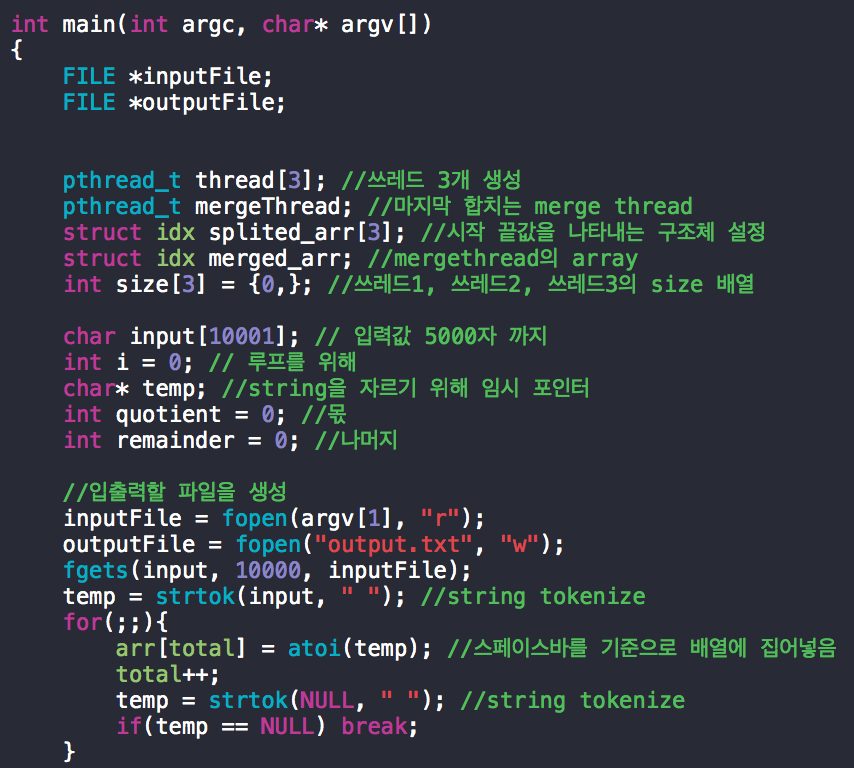
**시작값과 끝 인덱스를 가지고 있고,** 그리고 merge쓰레드인지 검사하는 struct를 만들었다. **전역배열 arr**과 전체 크기를 가지는 전역변수 total, 수행완료한 스레드의 갯수를 알리는 count를 설정했다.

<헤더파일>



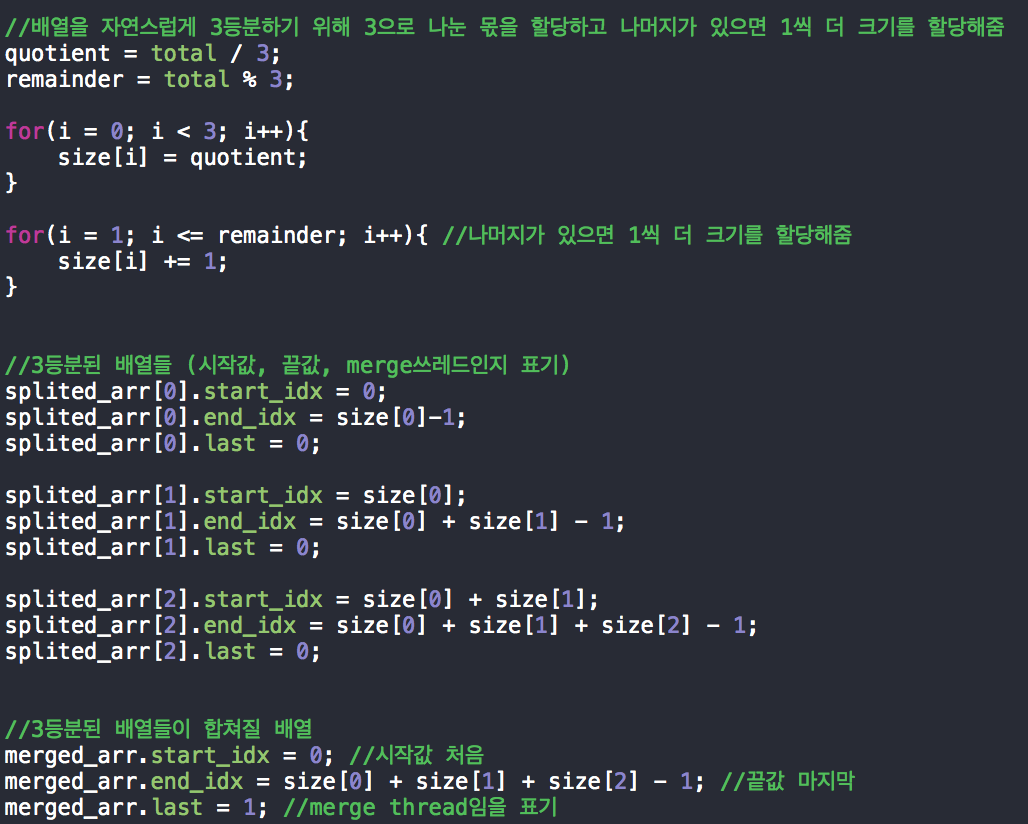
파일입력 출력을 위해 파일관련 헤더, 문자열을 다루므로(파일입력때 문자열을 읽음) string관련 헤더, 쓰레드를 다루므로 **pthread.h**를 include했다.

<메인함수>



파일입출력을 다루므로 argc, argv 메인 아규먼트를 이용했다. **총 4개의 쓰레드**(정렬스레드1, 정렬스레드2, 정렬스레드 3, merge스레드 4)를 다루므로 정렬스레드 3개를 배열로, 병합쓰레드를 따로 선언하였다. FILE을 이용하여 fopen으로 파일을 열고 for문을 순회하며 파일을 읽어서 string tokenize하며 int 형으로 변환해 배열에 집어넣는다. 지금 현재 Arr배열에는 지금 현재 정렬되지 않은 값들이 들어가 있다.

<메인 함수중 배열을 고르게 하는 부분>



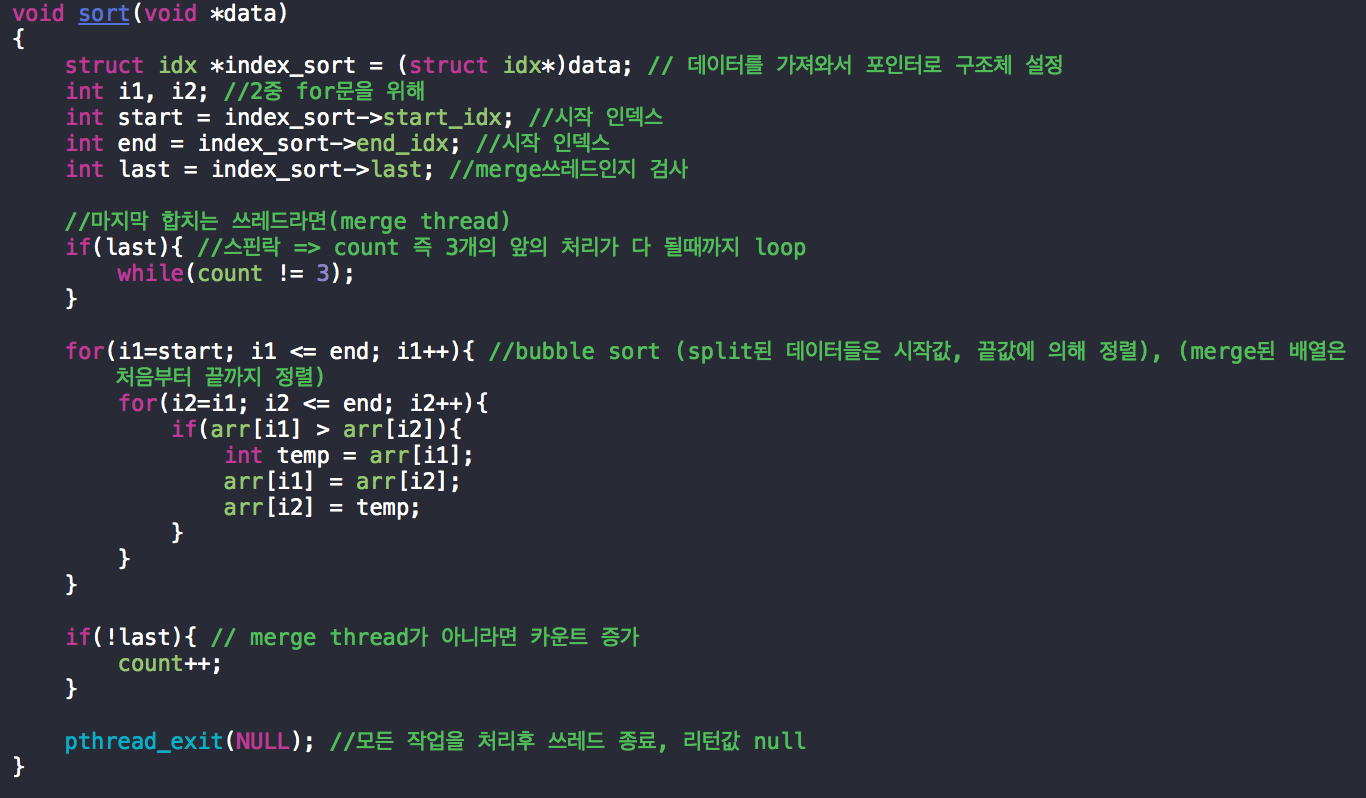
여기서 고른 배열이란 숫자가 7개 입력되었을때 2개,2개,3개 처럼 고르게 들어감을 말한다. 이를 나머지 연산을 통하여 각각 배열의 사이즈를 고르게 분배한다.

**시작값 끝값을 각각 사이즈에 맞추어 정해준다.** Last에는 이 값이 merge쓰레드 인지 알기위해 merge쓰레드에는 1을 넣어준다. 나머지 부분정렬 배열은 0을 넣는다.



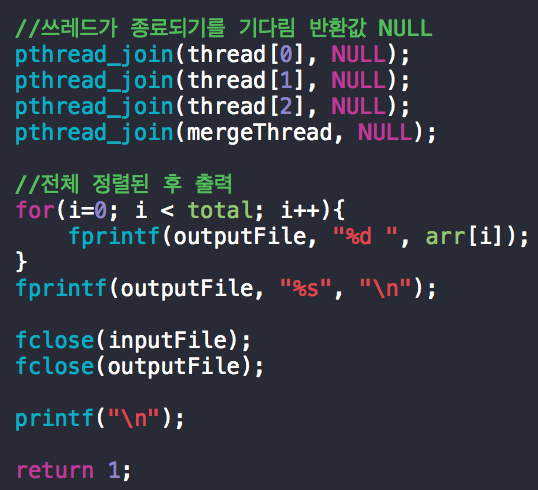
리눅스에서 사용하는 POSIX표준인 pthread\_create함수를 통하여 **쓰레드를 4개 생성한다.(정렬쓰레드 3, merge쓰레드 1)** 여기서 pthread\_create에서 에러가 발생하면 에러메시지를 출력하고 프로그램을 종료한다. Ptread\_create 함수는 첫번째 인자로 쓰레드를 받고, 두번째 인자로 옵션, 세번째 인자로 실행하는 함수, 번째 인자로 실행하는 함수의 인자를 받음을 알 수 있다.

<쓰레드 함수>



쓰레드 함수로서 void형의 sort함수를 만들었다. 이 쓰레드는 인자로 받아온 값을 struct형식으로 만든다**. 여기서 시작인덱스와 끝 인덱스**를 알아내 **bubble sort로 시작인덱스부터 끝 인덱스까지 정렬한다.** **이런 방법으로 data splitting을 한다.** 여기서 중요한 점은 **스핀락**을 이용하는 부분이다. Merge쓰레드가 아닌 정렬쓰레드1, 정렬쓰레드2, 정렬쓰레드 3는 if(last)문을 그냥 통과하기 때문에 상관없지만 merge쓰레드의 경우 if문에 걸려 while(count != 3)을 정렬쓰레드1,2,3들이 수행을 마칠때까지 계속적으로 수행되기 때문에 다른 쓰레드들이 마치기를 기다리고 있다. 정렬 쓰레드들은 bubble sort를 통해 시작과 끝인덱스를 통해 정렬하게되고, 정렬이 끝나면 count를 증가시키고 쓰레드를 exit하게 된다.(exit에서는 종료후 인자 전달을 할 수 있지만 여기서는 전달할 인자가 없으므로 NULL을 인자로 넣었다.) 여기서 눈여겨 볼 점은 쓰레드 1,2,3은 만들어진 순서대로 수행되는게 아니라 스케쥴러가 정한 순서대로 각각 병렬적으로 실행된다는 점이다. 다만 스핀락으로 정렬 1,2,3이 마치고 난 뒤에 마지막 merge 부분에서 전체적으로 정렬하는 코드가 마지막으로 실행된다는 점이 중요하다.

<다시 메인 함수내의 코드>

  
이렇게 정렬쓰레드1,2,3과,mergeThread가 모두 다 수행되기를 기다리고, 전체적으로 정렬된 배열 arr을 파일출력을 통해 출력하게 된다. 이 결과는 outout.txt에 저장된다.

1. **평가 및 개선방향**이렇게 멀티쓰레딩을 이용하여 정렬하는 application을 만들어 보았다. 처음 이용해보는 쓰레딩이었지만 pthread\_create, join, exit함수라는 posix표준 함수가 있었으므로 어렵지 않게 구현할 수 있었다. 구현의 편의성을 위해 전역배열에서 처리했지만 조금 더 심화해 본다면 전역배열에서만 처리하는게 아니라 전역배열에서 처리하고 정말 물리적으로 split (또다른 전역 배열에서) 하고 malloc을 이용해 동적으로 할당하는 프로그램을 만들어 볼 수 있을 것 같다. pthread\_create, join, exit의 함수에 들어가는 인자들이 조금 이해가 가지 않아서 인터넷에서 함수의 원형, 예제를 검색하며 알아보았다. 여러 예제들을 보니 함수 이용방법을 조금씩 터득하게 되었다. 프로그램의 장점이라고 생각하는 부분은 3으로 나누어 떨어진 수를 입력받는게 아니라 7, 8처럼 나누어 떨어지지 않는 수를 입력받아도 그 갯수에 맞게 (2,2,3이나 2,3,3)처럼 잘 갯수를 분배하는 코드가 장점이고 단점은 입력 갯수가 아주 많아졌을때 버블소트보다 좋은 성능의 소트가 있는데도 불구하고 버블소트를 사용했다는 점이다.
2. **기타 특이사항  
   윈도우에서는 POSIX표준이 아니므로 pthread를 이용할 수 없지만 맥(os X) 환경은 유닉스 기반이므로 pthread를 gcc로 컴파일 할 수 있기 때문에 실행이 가능했다. 또한 리눅스에 직접 올려서 작업해도 동일한 결과를 얻을 수 있었다. (우분투, 및 리눅스서버에서 확인)  
   다만 맥에서 gcc를 이용해 컴파일할때 undefined reference to `pthread\_create' 라는 오류가 발생 할 수도 있었는데 구글링해보니 이것은 –lpthread옵션을 통해 해결이 가능하다.   
   (–lpthread 옵션은 POSIX 표준 쓰레드 라이브러리를 사용한다는 옵션이다.)**
3. **참고자료**- 쓰레드 개념 및 예제  
   <https://ko.wikipedia.org/wiki/POSIX_%EC%8A%A4%EB%A0%88%EB%93%9C>  
   [**http://www.joinc.co.kr/w/man/3/pthread\_create**](http://www.joinc.co.kr/w/man/3/pthread_create)[**http://www.joinc.co.kr/w/man/3/pthread\_join**](http://www.joinc.co.kr/w/man/3/pthread_join)[**http://www.joinc.co.kr/w/man/3/pthread\_exit**](http://www.joinc.co.kr/w/man/3/pthread_exit) **- gcc 에서 –lpthread옵션**[**http://stackoverflow.com/questions/23250863/difference-between-pthread-and-lpthread-while-compiling**](http://stackoverflow.com/questions/23250863/difference-between-pthread-and-lpthread-while-compiling)