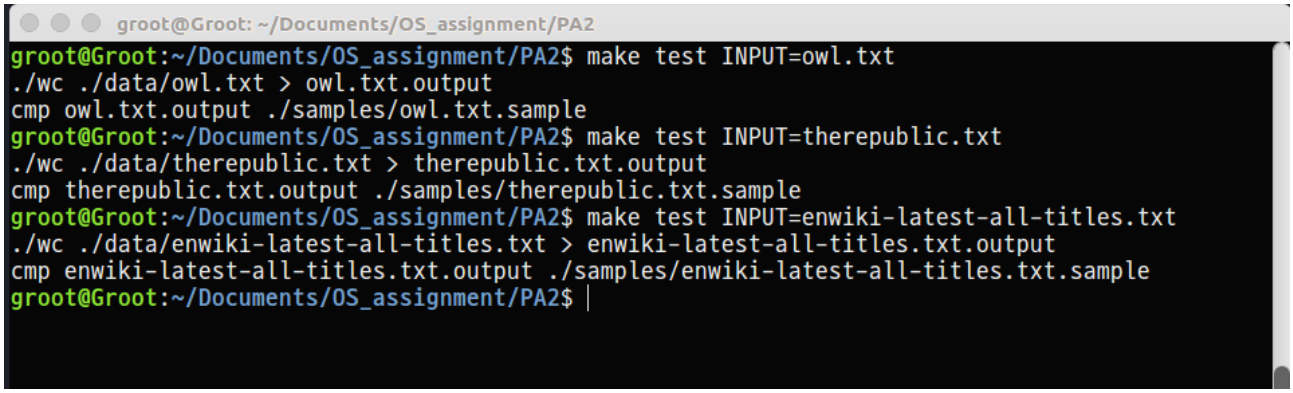
소프트웨어학과 201623405 금도윤

과제의 수행 정도

owl.txt 와 therepulic.txt, 그리고 위키, 파이썬 스크립트로 만든 텍스트 파일까지 시도해봤습니다. 결과는 모두 성공적이었고 자가 채점은 sample data 가 있는 3개만 하였습니다. (owl, therepublic, wiki)

결과는 sample 에 있는 파일과 동일합니다.



수행시간은 다음과 같습니다.

owl.txt

therepublic.txt

enwik.txt (약 3분 30초)

과제를 통해 배운 것

이번 과제를 통해 특히 많이 배운 것 같습니다...(딜레이 토큰을 모두 쓴만큼)

- 당연하지만 병렬 프로그래밍을 배웠습니다.
- 처음에 fgets 로 multithreading 을 하려 했으나 이 방법 보다는 파일 크기만큼 한 번에 읽는 것을 택하고 문자열 변환과 파싱에 multithreading 을 적용하기로 결정하였습니다.
- 행렬 곱셉을 병렬로 처리할 때 스레드 아이디를 배열의 index 로 사용해서 처리함을 보고 동기화를 유지하면서 병렬처리를 할 수 있다는 것을 알게 되었습니다. (보조 코드 정말 도움 많이 되었습니다.)
  이를 문자열 변환(특수문자 처리)에 사용하였습니다.
- 파싱(tokenizing 부분이 제일 힘들었습니다...) 처음에 while 문 조건으로 tok = strtok~~이런식으로 처리했다가 segmentation fault(core dumped)를 얼마나 많이 봤는지...위키를 테스트 할 때는 많은 시간과...고통이...심지어는 stack smashing detected 를 본적도 있습니다. 이런 오류를 거치면서 배열 크기에 매우 민감해졌습니다.
  결국은 기존의 strtok 로 문자열을 자르면 원본 문자열을 잃게되는 thread unsafety 와 달리, strtok_r 를 사용해서 thread 에서 safety 하게 단어만 뽑아낼 수 있었습니다. strtok_r 은 인자가 3개인데, 마지막 인자가 파싱 후 나머지 문자열을 저장하는 역할을 합니다. 이보다 각 함수의 내부를 살펴보면 static char*인 strtok, static 을 쓰지않는 strtok_r 을 통해서 왜 thread 에서 unsafe 하고 safe 한지 알 수 있었습니다.
  하지만 동기화가 문제였습니다. 동기화는 정말 설명할 수 없을 정도로 많은 시도 끝에 성공하였습니다. 조교님 말씀대로 직접 여러 번 해봐야 한다는 말씀이...진리였습니다. while 문 조건에 있던 strtok_r 를 빼고 strtok_r 를 따로 반복문 안에 넣어 파싱할 때의 mutex lock 과 파싱한 단어를 추가할 때 각각 mutex lock 을 걸어줌으로써 이를 해결하였습니다.

하지만 owl.txt 에서 갑자기 없는 !같은 문자열을 print 하게 되어 isalnum 으로 다시 한 번 문자나 숫자로 이루어진 문자열인지 판단하도록 하였습니다.
(strtok_r 은 c99에서 잘 수행이 안되기에(아예 수행이 안됩니다.) #define _GNU_SOURCE 를 추가해주었습니다.)

- 기존 sys/queue 의 수행시간을 측정해보니 수행 시간이 느리게 나오는 가장 큰 이유이기에 시간 복잡도가 보다 적은 hashTable 를 사용하였습니다. 실제 자료구조 시간에 이를 구현한 적은 있으나 문자열을 key 로 사용해 구현한 적은 없어 많은 부분을 참조하여 구현하였습니다. 검색해보니 확실히 java hashTable 에 대한 내용이 많이 나왔습니다.
  그 중 특이한건 hash function 을 구현할 때 문자열의 길이만큼 수행한다는 것었고, https://d2.naver.com/helloworld/831311에서 볼 수 있듯이 어떤 수에 (32=2^5)를 곱한 값은 shift 연산으로 쉽게 구할 수 있기에 해시 값이 같아질 확률을 낮추면서 소수인 31을 곱하는게 가장 효율적이라는 것을 알게되어 이를 사용하였습니다.
- HASHMAX(HashTable 크기)는 단어를 분류했을 때 총 개수보다 커야 하는데 약 2억1천으로 잡았습니다. 그 이상 잡으니 에러가 납니다. (왜 그런지 검색해도 안나옵니다...)
- qsort 함수 사용법을 알고만 있다가 직접 사용하게 되었습니다.
  Compare(정렬하기 위한 비교 함수)를 할 때 정렬 우선순위는 빈도수 > 사전 순이니까 처음에 내림차순, 빈도수 같으면 오름차순으로 정렬하도록 정의하였습니다.'
- Test 를 위해 thread id 와 sys/time.h 에서 시간 측정하는 방법도 알게 되었습니다.
- gdb 사용법을 익히고 디버깅을 해보았습니다. gdb wc 로 디버깅 시작 후 b 중단점 위치를 통해 확인하고 싶은 부분(중단점)을 결정하고 오류 찾고 상당히 유용했습니다.

과제에 대한 피드백

- HASHMAX 크기를 무작정 크게 잡는 바람에 메모리 측면에서 크나큰 낭비라고 볼 수 있습니다. 하지만 단어 수를 한 번에 알 수 없기에 다른 방법이 필요합니다.
- fread 로 파일 크기만큼 한 번에 읽기 때문에 파일이 훨씬 더 커진다면 문제가 생길 수 있습니다. 나눠서 병렬처리해야 할 수 있습니다.

*여담으로 report 작성을 위해 딜레이 토큰을 추가로 쓰게 되어 **다 쓰게 된 것은** 비밀입니다.