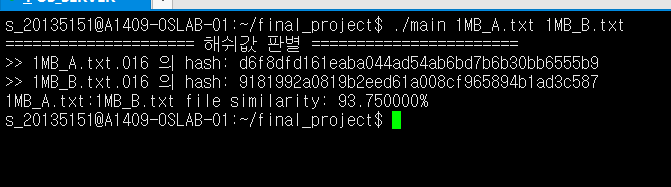
운영체제 Final\_Project

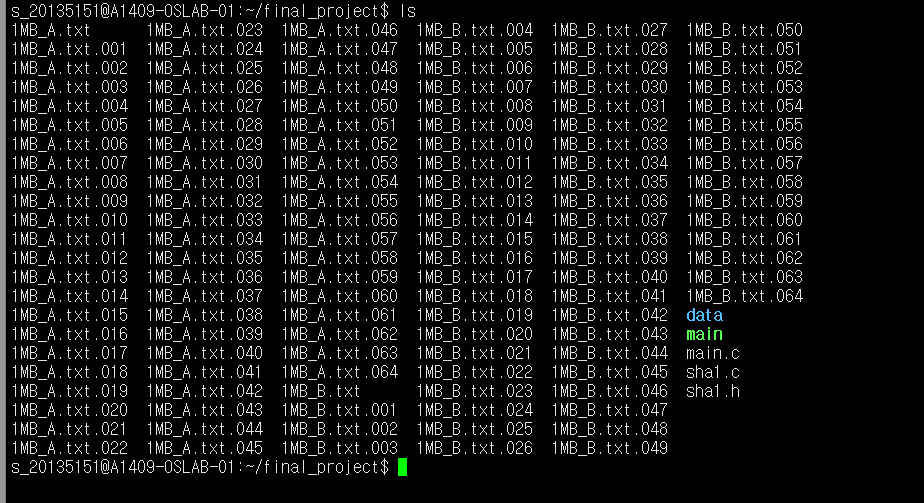
20135151 이갑성

[Basic]





[Intermediate]



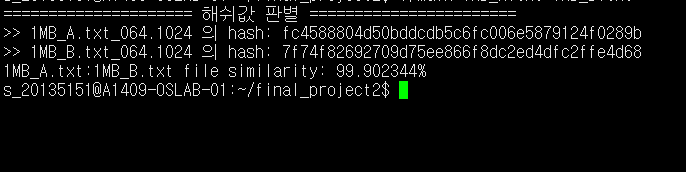


Basic에서 산출한 유사도: 93.75%

Intermediate에서 산출한 유사도: 98.4375%

다른 이유: Basic에서는 하나의 파일을 식빵으로 예를 들면 식빵을 16등분하여 그 중에서 다른 것(“썩은 것”으로 표현하겠다.) 하나를 찾는 것이다. 두 식빵을 비교해서 썩은 부분을 찾았다면 1/16의 범위 모두가 썩은 것이 아니기에 좀더 세밀한 찾아내는 작업이 필요하다. Intermediate에서는 식빵을 64등분하였기에 1/64이라는 좀더 세밀하게 썩은 부분을 찾을 수 있다.

[Advanced]



|  |
| --- |
| [소스코드]  #include <stdio.h>  #include <fcntl.h>  #include <sys/stat.h>  #include <sys/types.h>  #include <dirent.h>  #include <stdlib.h>  #include <unistd.h>  #include <string.h>  #include "sha1.h"  #include <time.h>  #define MAX 1024  int main(int argc, char \*argv[])  {  int i, n, j;  int fp, fp1, fp2;  char buffer[MAX];  SHA1Context sha1;  static unsigned char hash[20] = {0};  double similar;  int difNum = 0;  int n1, n2;  int fd1, fd2;  int chunkfd, nbytes;  char chunk\_name[128];  DIR \*dp;  struct dirent \*dirp;  char txtNameA[1024][20];  char txtNameB[1024][20];  int index = 0;  static unsigned char hashDataA[1024][100] = {0};  static unsigned char hashDataB[1024][100] = {0};  int dataAIndex = 0;  int dataBIndex = 0;  int result1, result2;  struct stat st;    int chunkN;  int CHUNKSIZE = 1024;  char chunkBuffer[CHUNKSIZE];  if(argc != 3)  {  perror("argc is not 3");  exit(0);  }  if((fd1 = open(argv[1], O\_RDONLY)) == -1){  perror("open failed");  exit(1);  }  if((fd2 = open(argv[2], O\_RDONLY)) == -1)  {  perror("open failed");  exit(1);  }    chunkN = 1;  while(nbytes = read(fd1, chunkBuffer, CHUNKSIZE)){  sprintf(chunk\_name, "%s.%03d", "1MB\_A.txt\_064", chunkN++);  strcpy(txtNameA[dataAIndex++], chunk\_name);  if((chunkfd = open(chunk\_name, O\_WRONLY | O\_CREAT, 0666)) == -1){  printf("read\_and\_chunk(): chunk file creat failed \n");  return -1;  }  write(chunkfd, chunkBuffer, CHUNKSIZE);  close(chunkfd);  }  chunkN = 1;  while(nbytes = read(fd2, chunkBuffer, CHUNKSIZE)){  sprintf(chunk\_name, "%s.%03d", "1MB\_B.txt\_064", chunkN++);  strcpy(txtNameB[dataBIndex++], chunk\_name);  if((chunkfd = open(chunk\_name, O\_WRONLY | O\_CREAT, 0666)) == -1){  printf("read\_and\_chunk(): chunk file creat failed \n");  return -1;  }  write(chunkfd, chunkBuffer, CHUNKSIZE);  close(chunkfd);  }  for(int j = 0; j < 1024; j++){  if((fp = open(txtNameA[j], O\_RDONLY, 0644)) == -1){  perror("open failed");  exit(1);  }  SHA1Reset(&sha1);  while((n = read(fp, buffer, MAX))){  SHA1Input(&sha1, buffer, n);  }  SHA1Result(&sha1, hash);  //해쉬값 복사작업  dataAIndex = 0;  for(i = 0; i < 20; i++)  {  hashDataA[dataAIndex][i] = hash[i];  }  close(fp);  }  for(int j = 0; j < 1024; j++){  if((fp = open(txtNameB[j], O\_RDONLY, 0644)) == -1){  perror("open failed");  exit(1);  }  SHA1Reset(&sha1);  while((n = read(fp, buffer, MAX))){  SHA1Input(&sha1, buffer, n);  }  SHA1Result(&sha1, hash);  //해쉬값 복사작업  dataBIndex = 0;  for(i = 0; i < 20; i++)  {  hashDataB[dataBIndex][i] = hash[i];  }  close(fp);  }  printf("===================== 해쉬값 판별 ======================= \n");  for(i = 0; i < 1024; i++)  {  for(j = 0; j < 20; j++){  if(hashDataA[i][j] != hashDataB[i][j]){  result1 = i;  break;  }  }  result2 = i;  }  printf(">> %s 의 hash: ", txtNameA[result2]);  for(int k = 0; k < 20; k++){  printf("%02x", hashDataA[result1][k]);  }  printf("\n");  printf(">> %s 의 hash: ", txtNameB[result2]);  for(int k = 0; k < 20; k++){  printf("%02x", hashDataB[result1][k]);  }  printf("\n");  similar = (((double)(CHUNKSIZE-1)/(double)CHUNKSIZE)) \* 100;  printf("1MB\_A.txt:1MB\_B.txt file similarity: %f%% \n", similar);  close(fd1);  close(fd2);  exit(0);  } |