1. 소스코드

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <unistd.h>  #include <sys/times.h>  #include <sys/wait.h>  void ssu\_do\_cmd(char \*cmd);  void ssu\_print\_times(clock\_t real, struct tms \*tms\_start, struct tms \*tms\_end);  void ssu\_echo\_exit(int status);  int main(int argc, char \*argv[]) {  int i;  //Non Buffering  setbuf(stdout, NULL);    //사용자로부터 명령어를 입력받아서 실행  for(i = 1; i < argc; i++)  ssu\_do\_cmd(argv[i]);  exit(0);  }  //사용자로부터 입력받은 명령어를 실행시키는 함수  void ssu\_do\_cmd(char \*cmd) {  struct tms tms\_start, tms\_end;  clock\_t start, end;  int status;  printf("\ncommand: %s\n", cmd);  //명령어 실행 전 클록 시간 반환  if((start = times(&tms\_start)) == -1) {  fprintf(stderr, "times error\n");  exit(1);  }  //명령어 실행  if((status = system(cmd)) < 0) {  fprintf(stderr, "system error\n");  exit(1);  }  //명령어 실행 후 클록 시간 반환  if((end = times(&tms\_end)) == -1) {  fprintf(stderr, "times error\n");  exit(1);  }    ssu\_print\_times(end - start, &tms\_start, &tms\_end);  ssu\_echo\_exit(status);  }  void ssu\_print\_times(clock\_t real, struct tms \*tms\_start, struct tms \*tms\_end) {  static long clocktick = 0;  if(clocktick == 0)  //초당 clock tick의 수 반환  if((clocktick = sysconf(\_SC\_CLK\_TCK)) < 0) {  fprintf(stderr, "sysconf error\n");  exit(1);  }  //clock time  printf(" real: %7.2f\n", real / (double) clocktick);  //User CPU time  printf(" user: %7.2f\n",  (tms\_end->tms\_utime - tms\_start->tms\_utime) / (double) clocktick);  //System CPU time  printf(" sys: %7.2f\n",  (tms\_end->tms\_stime - tms\_start->tms\_stime) / (double) clocktick);  //자식 사용자의 CPU time  printf(" child user: %7.2f\n",  (tms\_end->tms\_cutime - tms\_start->tms\_cutime) / (double) clocktick);  //자식 시스템의 CPU time  printf(" child sys: %7.2f\n",  (tms\_end->tms\_cstime - tms\_start->tms\_cstime) / (double) clocktick);  }  //자식 프로세스 상태값에 따른 정보를 출력하는 함수  void ssu\_echo\_exit(int status) {  if(WIFEXITED(status))//자식 프로세스가 종료하면  printf("normal termination, exit status = %d\n",  WEXITSTATUS(status));//exit()의 하위 8비트 값 반환  else if(WIFSIGNALED(status))//자식 프로세스가 시그널을 받았으나 처리하지 않아 비정상 종료가 된 경우  printf("abnormal termination, signal number = %d%s\n",  WTERMSIG(status),//시그널 번호 반환  #ifdef WCOREDUMP//코어 파일  WCOREDUMP(status) ? " (core file genereated)" : "");  #else  "");  #endif  else if(WIFSTOPPED(status))//자식 프로세스가 현재 중지 상태인 경우  printf("child stopped, signal number = %d\n",  WSTOPSIG(status));  } |

2. 실행결과

