```
/*
 * Name: Zhenjiang Tian (BlazerID: ztian)
 * Assignment: Lab-10
 * To compile:  gcc -Wall -Wextra -O2 -std=c11 lab10_solution.c -o lab10
 * To run:      ./lab10 commands.txt
 */
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <time.h>
#include <string.h>
#include <fcntl.h>

void createarray(char *buf, char **array) {
        int i, count, len;
        len = strlen(buf);
        buf[len-1] = '\0'; /* replace last character (\n) with \0 */
        for (i = 0, array[0] = &buf[0], count = 1; i < len; i++) {
                if (buf[i] == ' ') {
                   buf[i] = '\0';
                   array[count++] = &buf[i+1];
                }
        }
        array[count] = (char *)NULL;
}

int main(int argc, char **argv) {
    pid_t pid;
    int status;
    char line[BUFSIZ], buf[BUFSIZ], *args[BUFSIZ];
    time_t t1, t2;

    if (argc < 2) {
        printf("Usage: %s <commands file>\n", argv[0]);
        exit(-1);
    }

    FILE *fp1 = fopen(argv[1],"r");
    if (fp1 == NULL) {
        printf("Error opening file %s for reading\n", argv[1]);
        exit(-1);
    }

    FILE *fp2 = fopen("output.log","w");
    if (fp2 == NULL) {
        printf("Error opening file output.log for writing\n");
        exit(-1);
    }

    while (fgets(line, BUFSIZ, fp1) != NULL) {
      strcpy(buf, line); /* save line read */
      createarray(line, args);
#ifdef DEBUG
      int i;
      printf("%s", buf);
      for (i = 0; args[i] != NULL; i++)
          printf("[%s] ", args[i]);
      printf("\n");
#endif
      time(&t1);
      pid = fork();
      if (pid == 0) { /* lab10 */
        pid_t cpid = getpid();
        char outname[64], errname[64];
        snprintf(outname, sizeof(outname), "%ld.out", (long)cpid);
        snprintf(errname, sizeof(errname), "%ld.err", (long)cpid);
        int fdout = open(outname, O_WRONLY | O_CREAT | O_TRUNC, 0644);
        if (fdout < 0) { perror("open(<pid>.out)"); _exit(-1); }

        int fderr = open(errname, O_WRONLY | O_CREAT | O_TRUNC, 0644);
        if (fderr < 0) { perror("open(<pid>.err)"); _exit(-1); }

        if (dup2(fdout, STDOUT_FILENO) < 0) { perror("dup2(stdout)"); _exit(-1); }
        if (dup2(fderr, STDERR_FILENO) < 0) { perror("dup2(stderr)"); _exit(-1); }

        close(fdout);
        close(fderr);

        execvp(args[0], args);
        perror("execvp");
        _exit(-1);
      } else if (pid > 0) { /* this is the parent process */
        printf("Child started at %s", ctime(&t1));
        printf("Wait for the child process to terminate\n");
        wait(&status); /* wait for the child process to terminate */
        time(&t2);
        printf("Child ended at %s", ctime(&t2));
        if (WIFEXITED(status)) { /* child process terminated normally */
          printf("Child process exited with status = %d\n", WEXITSTATUS(status));
        } else { /* child process did not terminate normally */
          printf("Child process did not terminate normally!\n");
          /* look at the man page for wait (man 2 wait) to determine
             how the child process was terminated */
        }
        buf[strlen(buf) - 1] = '\t'; /* replace \n included by fgets with \t */
        strcat(buf, ctime(&t1)); /* append start time to command with arguments */
        buf[strlen(buf) - 1] = '\t'; /* replace \n added by ctime at the end with \t */
        strcat(buf, ctime(&t2)); /* append end time */
        fprintf(fp2, "%s", buf);
        fflush(fp2);
      } else { /* we have an error */
        perror("fork"); /* use perror to print the system error message */
        exit(EXIT_FAILURE);
      }
    }

    fclose(fp1);
    fclose(fp2);
    printf("[%ld]: Exiting main program .....\n", (long)getpid());

    return 0;
}
```