問題：コマンド行引数で「環境変数とファイル名」の組を複数指定し、環境変数を変更した上で出力をファイルにリダイレクトし date を実行するプログラムを作りなさい。

リスト 1: 解答例 1 （基本的なバージョン）

```
#include <stdio.h>                                  // perror のため
#include <stdlib.h>                                 // exit のため
#include <unistd.h>                                 // fork, execve のため
#include <sys/wait.h>                               // wait のため
#include <fcntl.h>                                  // open のため

char *execpath="/bin/date";                         // date プログラムのパス
char *args[] = { "date", NULL };                    // date プログラムの argv
extern char **environ;                              // 自分の環境変数

int main(int argc, char *argv[], char *envp[]) {
  for (int i=1; i<argc; i=i+2) {                    // コマンド行引数の各組について
    int pid = fork();                               // 分身 (子プロセス) をつくる
    if (pid < 0) {                                  // fork に失敗した場合は
      perror(argv[0]);                              //   fork のエラーメッセージ
      return 1;                                     //   エラー終了する
    }
    if (pid != 0) {                                 // 親プロセスなら
      int status;                                   //   子プロセスの終了を待つ
      wait(&status);
    } else {                                        // 子プロセスなら
      if (putenv(argv[i]) < 0) {                    //   環境変数を変更する
        perror(argv[i]);                            //   putenv が失敗した場合
        return 1;                                   //   エラー終了する
      }
      if (argv[i+1]!=NULL) {                        //   ファイル名あればリダイレクト処理
        close(1);                                   //     標準出力をクローズし
        int fd = open(argv[i+1],                    //     ファイルをオープンする
                      O_WRONLY|O_CREAT|O_TRUNC, 0644);
        if (fd < 0) {                               //     open がエラーなら
          perror(argv[i+1]);                        //       エラーメッセージを出力し
          return 1;                                 //       エラー終了する
        } else if (fd != 1) {
          fprintf(stderr, "リダイレクト失敗\n");    //     原因不明のエラーなら
          return 1;                                 //       エラー終了する
        }
      }
      execve(execpath, args, environ);              // date プログラムに変身する
      // ここが実行されるなら execve でエラーが発生(変身に失敗)
      perror(execpath);                             // エラーメッセージを出力して
      return 1;                                     // エラー終了する
    }
  }
  return 0;
}

/* 実行例
$ kadai TZ=Singapore a.txt TZ=Cuba c.txt
$ cat a.txt c.txt
Tue Jul 19 22:19:55 SGT 2016
Tue Jul 19 10:19:55 CDT 2016
*/
```

リスト 2: 解答例２（date 実行を関数にしたバージョン）

```
#include <stdio.h>                                  // perror のため
#include <stdlib.h>                                 // exit のため
#include <unistd.h>                                 // fork, execve のため
#include <sys/wait.h>                               // wait のため
#include <fcntl.h>                                  // open のため

char *execpath="/bin/date";                         // date プログラムのパス
char *args[] = { "date", NULL };                    // date プログラムの argv
extern char **environ;                              // 自分の環境変数

// 環境変数の変更とリダイレクトを行った上で date を実行する関数
//   仮引数 env は環境変数の変更指示をする文字列
//   仮引数 file は標準出力をリダイレクトするファイルの名前
void exeDate(char *env, char *file) {
  if (putenv(env) < 0) {                            // 環境変数を変更する
    perror(env);                                    // putenv が失敗した場合
    return;                                         //   エラーメッセージを出力して終了
  }
  // ファイル名がある場合のみリダイレクトする
  if (file!=NULL) {                                 // ファイル名があれば
    close(1);                                       //   標準出力をクローズし
    int fd = open(file, O_WRONLY|O_CREAT|O_TRUNC,   //   ファイルをオープンする
                  0644);
    if (fd < 0) {                                   //   open がエラーなら
      perror(file);                                 //     エラーメッセージを出力して終了
      return;
    } else if (fd != 1) {                           //   原因不明のエラーなら
      fprintf(stderr, "リダイレクト失敗\n");        //     エラーメッセージを出力して終了
      return;
    }
  }
  execve(execpath, args, environ);                  // date プログラムに変身する
  // ここが実行されるなら execve でエラーが発生(変身に失敗)
  perror(execpath);                                 // エラーメッセージを出力して終了
}

// execDate を分離したので main が小さくなった(読みやすいはず)
int main(int argc, char *argv[], char *envp[]) {
  for (int i=1; i<argc; i=i+2) {                    // コマンド行引数の各組について
    int pid = fork();                               // 分身 (子プロセス) をつくる
    if (pid < 0) {                                  // fork に失敗した場合は
      perror(argv[0]);                              //   fork のエラーメッセージを出力し
      return 1;                                     //     エラー終了する
    }
    if (pid != 0) {                                 // 親プロセスなら
      int status;                                   //   子プロセスの終了を待つ
      wait(&status);
    } else {                                        // 子プロセスなら date を実行する
      exeDate(argv[i], argv[i+1]);                  //   argv[i]="VAR=VAL"
      return 1;                                     //   argv[i+1]=ファイル名
    }                                               //   exeDate が返るならエラー終了する
  }
  return 0;                                         // 正常終了
}
```