An online judge system

The judge system uses fork() to create a child process.

In the child process, three things are done:
First, freopen() is invoked to redirect standard input / output to the files.
Secondly, setrlimit() is used to limit the time and output file size of the process. The hard limit is set larger than the soft limit to avoid a **SIGKILL** signal sent to the process before the desired signal is sent. We do not limit the memory usage, because a violation of this limit causes a **SIGSEGV** to be sent, which is not distinguishable from RUN-ERROR.

- When the output size limit is exceeded, a **SIGXFSZ** signal would be generated.
- When the time limit is exceeded, a **SIGXCPU** signal would be generated.

Finally, execl() is called, and the tested program is executed.

In the parent process, wait3() is used to wait for the child process to terminate and get the resource usage of it at the same time.
We first check whether a signal caused the process to terminate, and RUN-ERROR, FILE-LIMIT, TIME-LIMIT can be detected. If not, all resource the child process used is computed, and the limits are checked one by one.
If all limits are not violated, we check the output file with the correct output. This is done by the cmp() function, it compares the two files one character by one character, and returns 0 if a mismatch is found, otherwise, it returns 1;

Finally, the result is printed to the screen.

```
#include <unistd.h>
#include <sys/resource.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/wait.h>
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
typedef unsigned long sz_t;
const char *RES[6] = {"CORRECT", "WRONG-ANSWER", "RUN-ERROR", "TIME-LIMIT", "MEMORY-LIMIT", "FILE-LIMIT"};
enum {AC, WA, RE, TLE, MLE, FLE};
const char *JUDGE_OUT = "judge-output.shaform";
int cmp(FILE *a, FILE *b)
```

```
{
        int t;
        while ((t=fgetc(a)) == fgetc(b))
                if (t == EOF)
                        return 1;
        return 0;
}
sz_t get_file(const char *ef)
{
       struct stat st;
        if (stat(ef, &st) == -1)
                return 0;
       return (sz_t ) st.st_size;
}
void solve(const char *ef, const char *in, const char *out, int tl, int ml, int fl)
{
        int ret, res;
        pid_t c_pid;
        struct rusage ru;
        FILE *exout, *solout;
        sz_t mem, used, file;
        switch (c_pid = fork()) {
                case -1:
                        perror("fork failed!\n");
                        return;
                case 0:
                        /* child process */
                        freopen(in, "r", stdin);
                        freopen(JUDGE_OUT, "w", stdout);
                        struct rlimit lim;
                        // set time limit
                        lim.rlim_cur = tl+1;
                        lim.rlim_max = 2*lim.rlim_cur;
                        setrlimit(RLIMIT_CPU, &lim);
                        // set output limit
                        lim.rlim_cur = fl;
                        lim.rlim_max = 2*lim.rlim_cur;
                        setrlimit(RLIMIT_FSIZE, &lim);
                        // run !!
                        execl(ef, ef, NULL);
                        return;
                default:
```

```
                /* parent process */
                wait3(&ret, 0, &ru);
                if (WIFSIGNALED(ret)) {
                        switch (WTERMSIG(ret)) {
                                case SIGXFSZ:
                                        res = FLE;
                                        break;
                                case SIGXCPU:
                                        res = TLE;
                                        break;
                                default:
                                        res = RE;
                        }
                } else {
                        exout = fopen(JUDGE_OUT, "r");
                        solout = fopen(out, "r");
                        mem = ru.ru_minflt*getpagesize()/1024;
                        used = ru.ru_utime.tv_sec*1000000 + ru.ru_utime.tv_usec
                                + ru.ru_stime.tv_sec*1000000 + ru.ru_stime.tv_usec;
                        used /= 1000;
                        file = get_file(JUDGE_OUT);

                        if (mem > ml)
                                res = MLE;
                        else if (file > fl)
                                res = FLE;
                        else if (cmp(exout, solout))
                                res = AC;
                        else
                                res = WA;
                        fclose(solout);
                        fclose(exout);
                }
                if (res == AC) {
                        printf("CORRECT: %lu %lu %lu\n", used, mem, file);
                } else {
                        printf("%s\n", RES[res]);
                }
                fflush(stdout);
        }
}
const size_t BSZ = NAME_MAX+1;
```

```
int main()
{
	char ef[BSZ], in[BSZ], out[BSZ];
	int tl, ml, fl;
	while (scanf("%s %s %s %d %d %d", ef, in, out, &tl, &ml, &fl) == 6) {
		solve(ef, in, out, tl, ml, fl);
	}
	return 0;
}
```