Appendix 3—Safe Alternate Source Code to prevent TOCOTOU

race_condition.c

```c
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdlib.h>

#define HASH_OF_CORRECT_PASSWORD 0x4b665b0
#define BUFFER_LEN 48

void winner_handler() {
    printf("\033[32mCongraz! You complete the task!\033[0m\n");
    exit(0);
}

unsigned int getHash(const char* str, unsigned int len){
    const unsigned int bitsOfUnsignedInt = (unsigned int)(sizeof(unsigned int) * 8);
    const unsigned int threeQuarters = (unsigned int)((bitsOfUnsignedInt * 3)/4);
    const unsigned int halfQuarter = (unsigned int)(bitsOfUnsignedInt/8);
    const unsigned int highBits = (unsigned int)(0xFFFFFFFF <<(bitsOfUnsignedInt - halfQuarter));

    unsigned int hash = 0;
    unsigned int test = 0;

    for(int i = 0; i < len; ++i){
        hash = (hash << halfQuarter) + (*str++);

        if((test = hash & highBits) != 0){
            hash = ((hash^(test >> threeQuarters)) & (~highBits));
        }
    }

    return hash;
}
```

```c

void setup() {
    setvbuf(stdin, NULL, _IONBF, 0);
    setvbuf(stdout, NULL, _IONBF, 0);
    setvbuf(stderr, NULL, _IONBF, 0);
}

unsigned int get_file_size(const char *file) {
    struct stat statbuf;
    stat(file, &statbuf);
    int size = statbuf.st_size;
    return size;
}

int main() {
    setup();

    char user_name[BUFFER_LEN] = {0};
    char password[BUFFER_LEN] = {0};
    char flags[BUFFER_LEN] = {0};

    sleep(10); // Ideally this could be removed, but I imagine this is mimicking an expensive function, network or I/O call

    user_name_fd = open("./user_name.txt", O_RDWR);
    password_fd = open("./password.txt", O_RDWR);

    char c;
    int i = 0;
    for (i = 0; i < BUFFER_LEN - 1; i++) {
        read(user_name_fd, &c, 1);
        if ((c == '\n') || (c == -1)) break;
        user_name[i] = c;
    }
    printf("Read %d bytes from user_name.txt\n", i);

    for (i = 0; i < BUFFER_LEN - 1; i++) {
        read(password_fd, &c, 1);
        if ((c == '\n') || (c == -1)) break;
        password[i] = c;
    }
    close(user_name_fd);
```

```
    close(password_fd);
    printf("Read %d bytes from password.txt\n", i);
    printf("user_name ==> %s\n", user_name);
    printf("password ==> %s\n", password);

    if (strncmp(user_name, "We1s0n", 6) == 0 &&
        getHash(password, 32) == HASH_OF_CORRECT_PASSWORD) {
        strncpy(flags, "AcCes50k", 8);
    }

    if (strncmp(flags, "AcCes50k", 8) == 0) {
        printf("[+] Login Success!\n");
        winner_handler();
    }
    else {
        printf("[-] Wrong user name or password\n");
    }
}

// gcc -g ./race_condition.c -o ./race_condition
```