Learning Diary:

Morse Code:

Most important thing that I learned is how to use pipes. These are half duplex pipes. The child process converts each character into equivalent morse code string and puts it into the pipe. I got stuck when at the parent end, I didn't know how much data to recieve. To solve this, I used two pipes: one for sending data and another one for sending the size of the data. Knowing the size is needed because I made a string of appropiate size on the parent end. Newline is also handled.

Cleaning:

When I first started doing the program, I didn't notice the the “empty line“ requirement. Without it, I solved the “comments“ problem. Then when I noticed the “empty line“ requirement, I had to change the way I read the files. Now I read one line at a time and detected the presence of empty lines. But there is a flaw. I replaced the comment with spaces. As I noticed the “empty line“ requirement at the last moment, I could not fix this. As a result, the program rejects empty lines when reading the file, but replaces comments with spaces.

File processing:

Unfortunately, I am not submitting this problem.

Hello World:

The main problem here was “what to do“. What I did was, after forking, the child process sleeps while parent registers for SIGALRM. Then the child sends a SIGALRM to its parent. The function that handles the SIGALRM first finds the current directory and then searches the directory (not recursive) for the source file itself. Hence the progarm always prints “HELLO WORLD“.

Performance Evaluation:

When I started doing this program, I didn’t imagine it to be the most difficult. The basic idea is that, I read a large enough text file, located a particular string to replace and then write the data into another file. As it was not mentioned in the problem statement, I did not uses multiple processes or signals in this program

One of the programs (asyn_io) uses asynchronous or non blocking I/O. Common I/O operations, such as read(), fread() or write() functions are used in the other program (reg_io). Common I/O operations block the whole program until I/O operation is finished. Non blocking I/O on the other hand, uses aio_read() and aio_write() functions. The aio_read() and aio_write() functions return control to the calling process once the I/O is initiated, rather than after the I/O operation is complete.

The first hurdle for me was to use string library functions properly, so that string could be found and replaced efficiently. For this, I had to learn to use strstr() and strncpy() functions.

When measuring time, regular I/O program gives:

```
root@jars-desktop:/home/C_programming/82372J-05# time ./reg_io > dump

real    0m0.073s
user    0m0.048s
sys     0m0.020s
```

And when measuring time, asynchronous I/O program gives:

```
root@jars-desktop:/home/C_programming/82372J-05# time ./asyn_io >
dump
real    0m0.578s
user    0m0.016s
sys     0m0.020s
```

We know that “real“ time is the combination of of the amount of time the CPU spends performing some action for a program and the amount of time the CPU spends performaing system calls for the kernal on the program’s behalf. So, we see that, asynchronous I/O spends less time than regular I/O for the same amount of data. Though in both cases, system CPU time is the same.

First, I took a large enough text 122860 bytes long text file (called input). When I read the file into a string and manipulate it (i.e. find and replace) there is no problem. The outputs in the stdout shows this (I also poured the output in a file called “dump“). But when I try to write back this huge string into another file, I faced serious problems. In case of regular I/O program, I get weird characters where string is replaced. But when I print that huge string, everything is OK and there are no weird characters. I don’t have any previous experience in dealing with large text files in C. Maybe for this, I couldn’t solve this problem.

I used a tool called Valgrind, to detect errors in memory management. The output looks like:

```
root@ubuntu1004desktop:/home/C_programming/more_test/82372J-05#
valgrind ./reg_io
==7173== Memcheck, a memory error detector
==7173== Copyright (C) 2002-2010, and GNU GPL'd, by Julian Seward et al.
==7173== Using Valgrind-3.6.0 and LibVEX; rerun with -h for copyright info
==7173== Command: ./reg_io
==7173==
==7173== Conditional jump or move depends on uninitialised value(s)
==7173==    at 0x4025E0F: strcat (mc_replace_strmem.c:176)
==7173==    by 0x804887B: main (in
/home/C_programming/more_test/82372J-05/reg_io)
==7173==
==7173== Invalid write of size 1
==7173==    at 0x4025E3B: strcat (mc_replace_strmem.c:176)
==7173==    by 0x804887B: main (in
/home/C_programming/more_test/82372J-05/reg_io)
==7173==  Address 0x41b3015 is 0 bytes after a block of size 122,861 alloc'd
==7173==    at 0x4025000: malloc (vg_replace_malloc.c:236)
==7173==    by 0x8048822: main (in
/home/C_programming/more_test/82372J-05/reg_io)
==7173==
==7173== Invalid write of size 1
```

```
==7173==    at 0x4025E57: strcat (mc_replace_strmem.c:176)
==7173==    by 0x804887B: main (in
/home/C_programming/more_test/82372J-05/reg_io)
==7173==  Address 0x41b303c is not stack'd, malloc'd or (recently) free'd
==7173==
--7173-- VALGRIND INTERNAL ERROR: Valgrind received a signal 11
(SIGSEGV) - exiting
--7173-- si_code=1;  Faulting address: 0x6889914C;  sp: 0x62a8de58

valgrind: the 'impossible' happened:
   Killed by fatal signal
==7173==    at 0x380351B2: vgPlain_arena_malloc (m_mallocfree.c:245)

sched status:
  running_tid=1

Thread 1: status = VgTs_Runnable
==7173==    at 0x4025000: malloc (vg_replace_malloc.c:236)
==7173==    by 0x8048739: find_replace_string (in
/home/C_programming/more_test/82372J-05/reg_io)
==7173==    by 0x80488CE: main (in
/home/C_programming/more_test/82372J-05/reg_io)


Note: see also the FAQ in the source distribution.
It contains workarounds to several common problems.
In particular, if Valgrind aborted or crashed after
identifying problems in your program, there's a good chance
that fixing those problems will prevent Valgrind aborting or
crashing, especially if it happened in m_mallocfree.c.

If that doesn't help, please report this bug to: www.valgrind.org

In the bug report, send all the above text, the valgrind
version, and what OS and version you are using.  Thanks.
```

Looking at the first part of Valgrind's output, I see that,7173 is the process ID. This part of the error report says that a write error has occurred at line number 176, in the function strcat(). The function strcat() is called by function main. There are a lot of similar errors, but due to my inexperience with large strings in C, I couldn't solve the problem.

In case of the asynchronous I/O, the problem is even more weird. Readind from the file into a string and then replacing it worked as planned. At first outputs were also OK. The output from Valgrind shows:

```
root@ubuntu1004desktop:/home/C_programming/more_test/82372J-05#
valgrind ./asyn_io
==7156== Memcheck, a memory error detector
==7156== Copyright (C) 2002-2010, and GNU GPL'd, by Julian Seward et al.
==7156== Using Valgrind-3.6.0 and LibVEX; rerun with -h for copyright info
==7156== Command: ./asyn_io
==7156==
==7156== Conditional jump or move depends on uninitialised value(s)
==7156==    at 0x40B682F: ??? (in /lib/tls/i686/cmov/libc-2.11.1.so)
==7156==    by 0x40CE75B: ??? (in /lib/tls/i686/cmov/libc-2.11.1.so)
==7156==    by 0x40CD617: ??? (in /lib/tls/i686/cmov/libc-2.11.1.so)
==7156==    by 0x40CD720: ??? (in /lib/tls/i686/cmov/libc-2.11.1.so)
```

```
==7156==    by 0x40CBE5B: localtime_r (in /lib/tls/i686/cmov/libc-2.11.1.so)
==7156==    by 0x410C734: __vsyslog_chk (in /lib/tls/i686/cmov/libc-2.11.1.so)
==7156==    by 0x410CC76: syslog (in /lib/tls/i686/cmov/libc-2.11.1.so)
==7156==    by 0x80489A4: read_file (asyn_io.c:74)
==7156==    by 0x8048D21: main (asyn_io.c:215)
==7156==
==7156== Conditional jump or move depends on uninitialised value(s)
==7156==    at 0x40B682F: ??? (in /lib/tls/i686/cmov/libc-2.11.1.so)
==7156==    by 0x40CE7C4: ??? (in /lib/tls/i686/cmov/libc-2.11.1.so)
==7156==    by 0x40CD617: ??? (in /lib/tls/i686/cmov/libc-2.11.1.so)
==7156==    by 0x40CD720: ??? (in /lib/tls/i686/cmov/libc-2.11.1.so)
==7156==    by 0x40CBE5B: localtime_r (in /lib/tls/i686/cmov/libc-2.11.1.so)
==7156==    by 0x410C734: __vsyslog_chk (in /lib/tls/i686/cmov/libc-2.11.1.so)
==7156==    by 0x410CC76: syslog (in /lib/tls/i686/cmov/libc-2.11.1.so)
==7156==    by 0x80489A4: read_file (asyn_io.c:74)
==7156==    by 0x8048D21: main (asyn_io.c:215)
==7156==
==7156== Conditional jump or move depends on uninitialised value(s)
==7156==    at 0x40B682F: ??? (in /lib/tls/i686/cmov/libc-2.11.1.so)
==7156==    by 0x40CDC23: ??? (in /lib/tls/i686/cmov/libc-2.11.1.so)
==7156==    by 0x40CD7C2: ??? (in /lib/tls/i686/cmov/libc-2.11.1.so)
==7156==    by 0x40CBE5B: localtime_r (in /lib/tls/i686/cmov/libc-2.11.1.so)
==7156==    by 0x410C734: __vsyslog_chk (in /lib/tls/i686/cmov/libc-2.11.1.so)
==7156==    by 0x410CC76: syslog (in /lib/tls/i686/cmov/libc-2.11.1.so)
==7156==    by 0x80489A4: read_file (asyn_io.c:74)
==7156==    by 0x8048D21: main (asyn_io.c:215)
==7156==

 Request queued!
 Success!
==7156== Invalid write of size 1
==7156==    at 0x4025E3B: strcat (mc_replace_strmem.c:176)
==7156==    by 0x8048CA3: find_replace_string (asyn_io.c:191)
==7156==    by 0x8048D4C: main (asyn_io.c:217)
==7156==  Address 0x41da2b3 is 0 bytes after a block of size 59 alloc'd
==7156==    at 0x4025000: malloc (vg_replace_malloc.c:236)
==7156==    by 0x8048C58: find_replace_string (asyn_io.c:184)
==7156==    by 0x8048D4C: main (asyn_io.c:217)
==7156==
==7156== Invalid write of size 1
==7156==    at 0x4025E57: strcat (mc_replace_strmem.c:176)
==7156==    by 0x8048CA3: find_replace_string (asyn_io.c:191)
==7156==    by 0x8048D4C: main (asyn_io.c:217)
==7156==  Address 0x41da2bc is 9 bytes after a block of size 59 alloc'd
==7156==    at 0x4025000: malloc (vg_replace_malloc.c:236)
==7156==    by 0x8048C58: find_replace_string (asyn_io.c:184)
==7156==    by 0x8048D4C: main (asyn_io.c:217)
==7156==
==7156== Invalid read of size 1
==7156==    at 0x4025E1B: strcat (mc_replace_strmem.c:176)
==7156==    by 0x8048CC6: find_replace_string (asyn_io.c:195)
==7156==    by 0x8048D4C: main (asyn_io.c:217)
==7156==  Address 0x41da2b3 is 0 bytes after a block of size 59 alloc'd
==7156==    at 0x4025000: malloc (vg_replace_malloc.c:236)
==7156==    by 0x8048C58: find_replace_string (asyn_io.c:184)
==7156==    by 0x8048D4C: main (asyn_io.c:217)
==7156==
==7156== Invalid write of size 1
==7156==    at 0x4025E3B: strcat (mc_replace_strmem.c:176)
```

```
==7156==    by 0x8048CC6: find_replace_string (asyn_io.c:195)
==7156==    by 0x8048D4C: main (asyn_io.c:217)
==7156==  Address 0x41da2bc is 9 bytes after a block of size 59 alloc'd
==7156==    at 0x4025000: malloc (vg_replace_malloc.c:236)
==7156==    by 0x8048C58: find_replace_string (asyn_io.c:184)
==7156==    by 0x8048D4C: main (asyn_io.c:217)
==7156==
==7156== Invalid write of size 1
==7156==    at 0x4025E57: strcat (mc_replace_strmem.c:176)
==7156==    by 0x8048CC6: find_replace_string (asyn_io.c:195)
==7156==    by 0x8048D4C: main (asyn_io.c:217)
==7156==  Address 0x41f8268 is not stack'd, malloc'd or (recently) free'd
==7156==
--7156-- VALGRIND INTERNAL ERROR: Valgrind received a signal 11
(SIGSEGV) - exiting
--7156-- si_code=1;  Faulting address: 0x7786C336;  sp: 0x62b3ce58

valgrind: the 'impossible' happened:
   Killed by fatal signal
==7156==    at 0x380351B2: vgPlain_arena_malloc (m_mallocfree.c:245)

sched status:
  running_tid=1

Thread 1: status = VgTs_Runnable
==7156==    at 0x4025000: malloc (vg_replace_malloc.c:236)
==7156==    by 0x40A2BF5: open_memstream (in /lib/tls/i686/cmov/libc-2.11.1.so)
==7156==    by 0x410C6F4: __vsyslog_chk (in /lib/tls/i686/cmov/libc-2.11.1.so)
==7156==    by 0x410CC76: syslog (in /lib/tls/i686/cmov/libc-2.11.1.so)
==7156==    by 0x8048B77: write_file (asyn_io.c:140)
==7156==    by 0x8048D6C: main (asyn_io.c:218)

Thread 2: status = VgTs_WaitSys
==7156==    at 0x4000832: ??? (in /lib/ld-2.11.1.so)
==7156==    by 0x41A7403: pthread_cond_timedwait@@GLIBC_2.3.2 (in
/lib/tls/i686/cmov/libpthread-2.11.1.so)
==7156==    by 0xA1CBAD7: ???


Note: see also the FAQ in the source distribution.
It contains workarounds to several common problems.
In particular, if Valgrind aborted or crashed after
identifying problems in your program, there's a good chance
that fixing those problems will prevent Valgrind aborting or
crashing, especially if it happened in m_mallocfree.c.

If that doesn't help, please report this bug to: www.valgrind.org

In the bug report, send all the above text, the valgrind
version, and what OS and version you are using.  Thanks.
```

The program worked at first (the ayn_ouput.txt shows this). Then it started crashing by showing the following:

```
*** glibc detected *** ./asyn_io: malloc(): memory corruption: 0x08fc49d0 ***
======= Backtrace: =========
/lib/tls/i686/cmov/libc.so.6(+0x6b591)[0x8ad591]
/lib/tls/i686/cmov/libc.so.6(+0x6e395)[0x8b0395]
/lib/tls/i686/cmov/libc.so.6(__libc_calloc+0xab)[0x8b170b]
```

```
/lib/tls/i686/cmov/libc.so.6(open_memstream+0x5d)[0x8a1c2d]
/lib/tls/i686/cmov/libc.so.6(__vsyslog_chk+0x75)[0x90b6f5]
/lib/tls/i686/cmov/libc.so.6(syslog+0x27)[0x90bc77]
./asyn_io[0x8048b78]
./asyn_io[0x8048d6d]
/lib/tls/i686/cmov/libc.so.6(__libc_start_main+0xe6)[0x858bd6]
./asyn_io[0x80487d1]
======= Memory map: ========
00110000-00125000 r-xp 00000000 08:01 131356
/lib/tls/i686/cmov/libpthread-2.11.1.so
00125000-00126000 r--p 00014000 08:01 131356
/lib/tls/i686/cmov/libpthread-2.11.1.so
00126000-00127000 rw-p 00015000 08:01 131356
/lib/tls/i686/cmov/libpthread-2.11.1.so
00127000-00129000 rw-p 00000000 00:00 0
005d7000-005de000 r-xp 00000000 08:01 131358
/lib/tls/i686/cmov/librt-2.11.1.so
005de000-005df000 r--p 00006000 08:01 131358
/lib/tls/i686/cmov/librt-2.11.1.so
005df000-005e0000 rw-p 00007000 08:01 131358
/lib/tls/i686/cmov/librt-2.11.1.so
00842000-00995000 r-xp 00000000 08:01 131342
/lib/tls/i686/cmov/libc-2.11.1.so
00995000-00996000 ---p 00153000 08:01 131342
/lib/tls/i686/cmov/libc-2.11.1.so
00996000-00998000 r--p 00153000 08:01 131342
/lib/tls/i686/cmov/libc-2.11.1.so
00998000-00999000 rw-p 00155000 08:01 131342
/lib/tls/i686/cmov/libc-2.11.1.so
00999000-0099c000 rw-p 00000000 00:00 0
00a54000-00a6f000 r-xp 00000000 08:01 131083     /lib/ld-2.11.1.so
00a6f000-00a70000 r--p 0001a000 08:01 131083     /lib/ld-2.11.1.so
00a70000-00a71000 rw-p 0001b000 08:01 131083     /lib/ld-2.11.1.so
00cb9000-00cba000 r-xp 00000000 00:00 0          [vdso]
00d0e000-00d2b000 r-xp 00000000 08:01 131127     /lib/libgcc_s.so.1
00d2b000-00d2c000 r--p 0001c000 08:01 131127     /lib/libgcc_s.so.1
00d2c000-00d2d000 rw-p 0001d000 08:01 131127     /lib/libgcc_s.so.1
08048000-08049000 r-xp 00000000 08:01 529955
/home/C_programming/more_test/82372J-05/asyn_io
08049000-0804a000 r--p 00001000 08:01 529955
/home/C_programming/more_test/82372J-05/asyn_io
0804a000-0804b000 rw-p 00002000 08:01 529955
/home/C_programming/more_test/82372J-05/asyn_io
08fa4000-08fe3000 rw-p 00000000 00:00 0          [heap]
b7600000-b7621000 rw-p 00000000 00:00 0
b7621000-b7700000 ---p 00000000 00:00 0
b7778000-b777a000 rw-p 00000000 00:00 0
b7783000-b7784000 rw-p 00000000 00:00 0
b7784000-b7785000 ---p 00000000 00:00 0
b7785000-b778a000 rw-p 00000000 00:00 0
bfb99000-bfbae000 rw-p 00000000 00:00 0          [stack]
Aborted
```

I assume that, the program is writing outside the bounds of a block of memory that it had malloc()ed.

Conclusion: Looking at this assignment, things like signals and IPCs did not pose that big of a problem (though this was my first time with these). The biggest challenge was handling large strings. My string manipulation logics worked as planned. But somehow storing that string and then dumping it

into file did not work as I had wished. The biggest lesson for me, large strings in C needs to be handled with care.