Midterm

This is a preview of the published version of the quiz

Started: Mar 9 at 11:27am

Quiz Instructions

Please do all questions.

Question 1 — 0 pts

Please enter your username. For example, my username is **jplank**.

Question 2 — 0 pts

If you feel compelled to write me a note about any of your answers, please do it here, rather than in the answer boxes of the various questions. You don't need to do this; however, I've learned that some students feel that it is necessary.

Edit View Insert Format Tools Table

12pt Paragraph B I U A T²

p

0 words

Question 3

25 pts

You are on a programming team for an embedded device that has a very primitive version of C/Unix as its programming environment. Your boss wants you to implement a procedure **write_ints()** that works like the following:

```
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>

int bit_set(const unsigned char *to_include, int index);

void write_ints(int fd, int size, const int *ints, const unsigned char *to_incl
ude)
{
  int i;

  for (i = 0; i < size; i++) {
    if (bit_set(to_include, i)) write(fd, &(ints[i]), sizeof(int));
  }
}
```

Here's the functionality in English: You treat **to_include** as an array of bits. **bit_set(to_include, i)** returns zero if the *i-th* bit of **to_include** is zero, and one if the *i-th* bit of **to_include** is one.

You want to write all of the integers in *ints*, corresponding to the set bits in *to_include* to the given file descriptor.

The implementation of **write_ints()** functions correctly, but it is way too slow. Your job is to:

- Tell me in two sentences why the implementation above is too slow.
- Implement **bet_set()**
- Implement **write_ints()** so that it is faster. Since you are on a restricted, embedded device, you don't have the stdio library available to you, nor do you

have **malloc()**. Your boss has told you that you may only use 1024 extra bytes of memory as a global variable, plus an additional 24 bytes of local variables.

Please remember that you can convert the "Paragraph" in your essay box to "preformatted", and you can resize it.

In case you're worried, the parameters of **write()** are:

```
write(int file_descriptor, void *bytes, unsigned long size);
```

Edit View Insert Format Tools Table

12pt Paragraph

p 0 words

Question 4 25 pts

Your good friend Luigi has written a program to remove all of the **a.out** files in all of his directories. It is included below. It doesn't work. Since Luigi is one of those people who would rather have interpersonal interactions than actually debug his code, he shows it to you so that you can help him.

You look at Luigi's code and ask him if he has taken CS360. He reports that he didn't go to UT, so, "no." Figures. I want you to tell me at least five places where Luigi has to

fix his code.

For each of these things, please do the following:

1. Say what the problem is and what line it is on.
2. Describe how it will manifest with an error.
3. Describe how you would fix it. You don't have to write code, but I don't want you to be vague.

By the way, this code compiles fine. For some relevant information:

- **opendir()/readdir()/closedir()** are all library calls that have been described in the lecture notes.
- **stat()** is a system call that has been described in the lecture notes. It returns 0 on success.
- **remove()** is a system call that "removes" the file in the given path like the **rm** command.
- **S_ISDIR()** is a macro that tests a bit of its parameter to see if it has been set. You treat it like a boolean.

```
/* 01 */  #include <stdio.h>
/* 02 */  #include <stdlib.h>
/* 03 */  #include <string.h>
/* 04 */  #include <dirent.h>
/* 05 */  #include <sys/stat.h>
/* 06 */
/* 07 */  void rm_a_dot_out(char *dir)
/* 08 */  {
/* 09 */      DIR *d;
/* 10 */      struct dirent *de;
/* 11 */      struct stat buf;
/* 12 */
/* 13 */      d = opendir(dir);
/* 14 */
/* 15 */      for (de = readdir(d); de != NULL; de = readdir(d)) {
/* 16 */          if (strcmp(de->d_name, "a.out") == 0) remove(de->d_name);
/* 17 */          if (stat(de->d_name, &buf) == 0) {
/* 18 */            if (S_ISDIR(buf.st_mode)) rm_a_dot_out(de->d_name);
/* 19 */           }
/* 20 */      }
/* 21 */      closedir(d);
/* 22 */  }
/* 23 */
/* 24 */  int main()
/* 25 */  {
/* 26 */      rm_a_dot_out(".");
/* 27 */      return 0;
/* 28 */  }
```

Edit View Insert Format Tools Table

12pt Paragraph

p

0 words

Question 5

9 pts

Suppose I do the following:

```
UNIX> mkdir d1
UNIX> cd d1
```

And now I do the following command:

```
UNIX> echo "My name is Harvey" > f1.txt
```

That command has modified three places on disk. Please tell me what they are and what information they store. Note, each answer is one line.

- Place #1:
- Place #2:
- Place #3:

Question 6 — 15 pts

For this question, please assume that your machine is little endian and has four-byte pointers.

I compile the program at the bottom of the question to **a.out** and then I run:

```
UNIX> ./a.out staph rust bo
```

When I do that, the first line of output is:

```
0x30 0x41 0x61 0x0
```

In the blanks below the program, please tell me lines two through six.

If you need syntax for some of these calls, here it is:

```
char *strcpy(char *dest, const char *src);
char *strcat(char *dest, const char *src);
void *memcpy(void *dest, const void *src, size_t n);
```

Here is the program:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char **argv)
{
  char s1[20], s2[20], s3[20];
  int *ip2, *ip3;
  int i, j;

  printf("0x%x 0x%x 0x%x 0x%x\n", '0', 'A', 'a', '\0');   /* First line */

  strcpy(s1, "0123456789012345");
  strcpy(s2, "0123456789012345");
  strcpy(s3, "0123456789012345");

  memcpy(s1+5, argv[1], strlen(argv[1]));
  printf("%s\n", s1);                                     /* Second line */

  ip2 = (int *) s2;
  memcpy(ip2+1, argv[2], strlen(argv[2])+1);
  printf("%s\n", s2);                                     /* Third line */

  ip3 = (int *) s3;
  printf("0x%x\n", ip3[2]);                               /* Fourth line */

  i = s3[5];
  j = s3[12];
  printf("0x%x\n", (i << 8) | j);                         /* Fifth line */

  i = s3[7];
  strcat((char *) &i, argv[3]);
  printf("%s\n", (char *) &i);                            /* Sixth line */

  return 0;
}
```

And here is where to put your answers:

- Line 2:
- Line 3:
- Line 4:
- Line 5:
- Line 6:

Question 7

16 pts

Our machine is little endian with four-byte pointers.

The following table shows the contents of memory starting at address 0x5b89fcac:

```
Addresses         Values
                 3 2 1 0
0x5b89fcac -- 0x5b89fcc8
0x5b89fcb0 -- 0x5b89fcbc
0x5b89fcb4 -- 0x5b89fcb0
0x5b89fcb8 -- 0x5b89fcb4
0x5b89fcbc -- 0x5b89fcb8
0x5b89fcc0 -- 0x5b89fcc0
0x5b89fcc4 -- 0x5b89fcc4
0x5b89fcc8 -- 0x5b89fcbc
0x5b89fccc -- 0x5b89fca3
0x5b89fcd0 -- 0x5b89fceb
0x5b89fcd4 -- 0x5b89fcd5
0x5b89fcd8 -- 0x5b89fcbc
0x5b89fcdc -- 0x5b89fcc3
0x5b89fce0 -- 0x5b89fca2
0x5b89fce4 -- 0x5b89fcef
0x5b89fce8 -- 0x5b89fcb0
```

Let us suppose that we have the following four pointers:

```
unsigned int *p;
unsigned int **q;
unsigned char *s;
unsigned char **t;
```

When we run the following piece of code, all four pointers equal 0x5b89fcac:

```
printf("0x%x\n", (*p)      & 0xff);       /* Line 1 */
printf("0x%x\n", (p[10])   & 0xff);       /* Line 2 */
printf("0x%x\n", (q[3][12])& 0xff);       /* Line 3 */
printf("0x%x\n", (q[0][3]) & 0xff);       /* Line 4 */
printf("0x%x\n", s[3]);                   /* Line 5 */
printf("0x%x\n", s[4]);                   /* Line 6 */
printf("0x%x\n", t[2][8]);                /* Line 7 */
printf("0x%x\n", t[1][16]);               /* Line 8 */
```

Please tell me the eight lines of output. If the answer is seg-fault or bus-error, then put that If the answer is unknown, then enter "unknown."

- Line 1:
- Line 2:

- Line 3:
- Line 4:
- Line 5:
- Line 6:
- Line 7:
- Line 8:

Saving... Submit Quiz