ICS Homework 9

May 9, 2020

1 Organization

1.1

Assume we compile the code:

```c
const int const_five = 5;
const int const_two = 2;
int five= 5;
int two = 0;
int uninitialized;
static int static_ten = 10;
static int static_uninit;
int f() {
    static f_static_i = 10;
    int f_i = 0;
}
int main() {
    return 0;
}
```

Please fill the following table:
(Note: show the distinction between .bss and COMM)

Symbol	.symtab entry?	Symbol type	Global/Local	Section
const_five				
const_two				
five				
two				
uninitialized				
static_ten				
static_uninit				
f				
f_static_i				
f_i				
main				

1.2

This problem concerns the m.o module from Figure 7.5 in the text book (csapp) and the following version of the swap.c function that counts the number of times it has been called:

```
extern int buf[];
int *bufp0 = &buf[0];
static int *bufp1;
static void incr(){
        static int count = 0;
        count++;
}

void swap(){
        int temp;
        incr();

        bufp1 = &buf[1];
        temp = *bufp0;

        *bufp0 = *bufp1;
        *bufp1 = temp;
}
```

For each symbol that is defined and referenced in swap.o , indicate if it will have a symbol table entry in the .symtab section in module swap.o . If so, indicate the module that defines the symbol (swap.o or m.o), the symbol type(local, global, or extern), and the section (.text, .data , or .bss) it occupies in that module.

Symbol	.swap.o .symtab entry?	Symbol type	Module where defined	Section
buf				
bufp0				
bufp1				
swap				
temp				
incr				
count				

1.3

Please show the relocation entry and process of the following instruction:

```
Disassembly of section .text:
0000000000000000 <main>:
```

```
           ...
           2: c7 05 00 00 00 00 ed ad 0f 00.
                                 movl$0xfaded, buf(%rip)
```

Relocation entry:

offset		type		symbol		addend	

Relocation:
ADDR(.text) = 0x4004d6
refaddr =
ADDR(r.symbol) = 0x601030
*refptr =

2 System Software

2.1

Assume we want to write a tick program which prints a "BEEP" in console every second. If there is any client connected, the BEEP will be sent to client instead of being printed on server. When client closes the connection, "BEEP" should be printed in console again. Here is part of the program:

```
void handler(int sig) {
  write(1, "BEEP\n", 5);
  alarm(1);
}

// This function will block and return
// after the client close the connection
void wait_disconnect(int fd) {
  char c;
  while (read(fd, &c, 1) > 0 || errno == EINIR);
}

int main(void) {
  int listenfd, connfd;
  // You are not allowed to use stderr
  close(2);
  signal(SIGALRM, handler);
  alarm(1);
  listenfd = open_listenfd(1234);
  while (1) {
    connfd = accept(listenfd, NULL, NULL);
    /* Your code here */
  }
  exit(0);
}
```

Please complete the program.