CS 342302 Operating Systems

Fall Semester 2021

Prof. Pai H. Chou

Weekly Review 3

The questions here serve the purpose of reviewing concepts from the lecture, and expect the concepts to be tested on the midterm and final. However, they are by no means exhaustive. Anything covered in the lecture and projects can be tested.

1. Definitions and Short Answers - week 3 (9/27 lectures)

1. A shell can implement all of the commands that it supports, as opposed to relying on external programs that are not part of the shells. What are the advantages and disadvantages?
2. Does the unix-style shell (such as bash, csh, etc) implement all of the commands that it can execute or not? Why is it a good idea?
3. Does the shell (GUI, CLI, etc) run in user mode or kernel mode? Why?
4. What is the purpose of a **system call**?
5. Why doesn't a system call instruction take the target address of the routine, unlike a jump or call instruction? How does it indicate **what service to request**?
6. What are ways **parameters** can be passed to a system call?
7. Are the functions in standard-I/O library <stdio.h> all system calls? For example, is printf() a system call? If not, how does it perform I/O operation? What actual system call does printf() call?
8. Do all **system calls** execute in kernel mode?
9. How are the functions in the **POSIX API** related to **system calls**?
10. Why would an application programmer prefer programming using an API than making system calls directly?
11. Do all <stdio.h> functions **make one or more system calls** in their implementation? Why or why not?
12. Does MS-DOS **create a new process** to run a user program? What happens to the shell when the user program is running and when it exits?
13. At what point does FreeBSD **start the shell**? Is there just one kind of shell?
14. How does a shell on FreeBSD **start a process**? What does the shell do when the process is executing?
15. What are the purposes of **system programs**?
16. What is the difference between **policy** and **mechanism**? What is the principle for the separation between policy and mechanism?
17. MS-DOS has a simple structure consisting of BIOS and DOS device drivers, resident system program, and application program. What are the advantages and disadvantages with this minimal structure?
18. Do **system programs** in Unix including shells, compilers, and interpreters run in kernel mode or user mode?
19. Is traditional Unix a **two-layer** or **N-layer** structure? What are its pros and cons?
20. In an N-layered OS structure, what layer is **hardware**? user interface? What is the dependency between a lower and an upper layer?
21. What are the advantages of layered approach? What is a successful example use? What are the disadvantages compared to the 2-layer, tightly coupled structure?
22. What is the key idea with **microkernels** compared to **monolithic** kernels?
23. Which of the following functions of a microkernel is done in kernel mode? in user mode?
	1. network driver
	2. device driver
	3. graphic driver
	4. interprocess communication
	5. CPU scheduling
	6. memory management
	7. file system
	8. application program
24. How can a microkernel run an OS service in **user mode** while also **protecting** the rest of the system?
25. Which of the following is easier or more efficient to do on a microkernel or a monolithic kernel? Why?
	1. port to a new architecture
	2. add new features
	3. OS overhead
26. What are the two main models for **interprocess communication**?
27. How does a modular OS divide its functionality? What are the advantages of this organization?
28. What are **loadable kernel modules** (LKM)? Do they run in the same **address space** as the kernel or different?
29. How would you characterize Linux and Solaris? monolithic or microkernel? combined LKM or hardcoded services in the kernel?
30. Is Darwin a monolithic or microkernel? What are the advantages of kexts in this context?
31. What is the purpose of the BSD Unix subsystem in Darwin?
32. Why does Android replace glibc with Bionic?
33. Since Android uses Linux kernel, does it mean it can also run executable programs for desktop and laptop Linux? Why or why not?
34. What is a **core dump** file? How is it different from a **log file**?
35. What is a **crash dump** and how is it different from a **core dump**?

2. EdSim51 and 8051 - week 3 (9/29 lecture)

1. What is the interrupt vector (address of ISR) of UART on 8051?
2. To use UART interrupt on the 8051, why is it it necessary to lay out the code memory this way:
 ORG 0H
 JMP Main
 ORG 23H
 JMP Serial\_ISR
Main: ...
Serial\_ISR: ...
3. The UART uses Timer-1 to generate its timing. What do the following bit registers do?

a. TR1

b. ET1

c. TF1

1. What is SDCC?
2. What is the size (in terms of bits or bytes) of the following types in SDCC? and are they signed or unsigned?

 .

a. bit

b. char

~~b. byte~~ (use #include<stdint.h> and uint8\_t instead) (refer to Q&A)

c. int

d. long

e. float

1. What does SDCC do if you invoke it with the following arguments?

a. sdcc file.c

b. sdcc -S file.c

c. sdcc -c file.c

d. sdcc file.rel -o file.ihx

1. Why does main in .c file get translated into \_main in the .asm file?
2. The .asm output from SDCC is different from Intel assembly syntax and therefore cannot be loaded directly into EdSim51 and run. So what file generated by SDCC (or related tools) in order to run on EdSim51?
3. In SDCC's <8051.h> header file, how is the GPIO port P1 **declared** so that it refers to the special-function register at memory address 90H?
4. Why does SDCC require that you declare an ISR with the \_\_interrupt keyword and a number such as (4) as in
void Serial\_ISR(void) \_\_interrupt(4) {
 ...
}
5. What is on the stack when an ISR is invoked?
6. Why do we NOT recommend **calling a function** such as DisplayLED(char num) from within the ISR?
7. By observing SDCC's assembly output for the ISR that calls a function (slides 24 and 26), what is the purpose of all the push (bits, acc, b, dpl, dph, 0+7, 0+6, … psw) instructions and all the corresponding pop instructions before the reti instruction?
8. What is the purpose of calling main() in the function definition
void \_sdcc\_gsinit\_startup(void) { main(); }
9. What happens if you don't define the following routines to empty when compiling and linking with SDCC?
void \_mcs51\_genRAMCLEAR(void) {}
void \_mcs51\_genXINIT(void) {}
void \_mcs51\_genXRAMCLEAR(void) {}
10. What clock frequency should be used when running the UART at 4800 baud?
11. How do you define a lookup table in SDCC to put data into CODE memory for mapping digits (0-9) to the LED segment to light up?
12. On slide 21, the \_sdcc\_gsinit\_startup() function contains inlined assembly code
\_\_asm
 mov sp, #0x57
\_\_endasm;
Can it be rewritten as C? How?

3. Short Assembly Programs

1. Try running the C program on slide 11. Make sure it compiles using the command on slide 15 and load the .ihx file into EdSim51 ("Load" button)
2. Try running the two-file example (LEDtest0.c, LEDseg.c) shown on slide 16, compile and link with the commands on the slide, and load into EdSim51 to run.
3. Try running the Serial Echo example on slide 18
4. Try running the Polling UART example on slide 19
5. Try running the intrLED.c example on slide 20. What is the difference between the use of Main() in previous examples and main() in this example?