Midterm 1.1

(Answers)

1. What is the problem with the following code?

int i  = 5

char ptr = &i;

printf(“%d\n”, i);

Ans: (1.5 points for identifying each error)

line 1 does not have semicolon.

line 2, we are dereferencing the pointer, which is counterproductive.

2. What are the sizes of arr and p in the C program in the 32-bit machine?

int arr[50];

int \*p = arr;

Ans: (1.5 points for each correct answer)

arr: 400 bytes

p:4 bytes

3. What does a pointer to an array contain?

Ans:

address of first element of an array

4. What is the potential address space of a system which uses 32-bit word size?

Ans:

2^32 or 4GB

5. How can we use a global variable that is defined in another .c/.h file?

?

Ans:

Use extern keyword

6. What is piping in the bash shell? Write a shell command using pipe(s).

Ans:

(3 pts)

Piping connects two commands in shell. The Unix systems allow stdout of a command to be connected to stdin of another command. You can make it do so by using the pipe character ‘|’

(2 pts)

Example: any correct command with piping

7. Give two disadvantages of compiling device drivers into the kernel.

Ans: (2.5 points for one mentioned disadvantage)

•required the administrator to re-compile the operating system   
when a new device type was introduced  
•each system had a different kernel

•Recompilation of the kernel is problematic  
 takes a long time  
 requires sophistication  
 versioning problems

8. Explain the term abstraction with one example.

Ans:

(3 pts)

Abstraction presents a complex thing in a simplest manner. In order to make a system user friendly to users, system provides abstraction of a complex details.

(3 pts)

Examples: can be real world example or from computer science field. An example of abstraction in computer science field is system calls provided by OS to user applications to deal with hardware. User applications don’t need to deal with nasty details of low-level implementation of computer. They use system calls whenever they need to access CPU, memory or any other resources.

9. Based on the code snippet, finish the function prototype for the *foo* function, clearly showing the type of each argument.

void foo(                                                                                               );

….

int main() {

  int \*a, c[10]={[0,2]=10};

  char  b;

  a = c;

  foo((float \*)&a[0], b, c);

}

Ans: (1.5 points for each correct declaration)

void foo( float \*a, char b, int \*c)

10. In the above question, what is the value of \*(a+3)? why?

\*(a+3) =0, (2pts)

because:

it is the same as c[4]; (1 pt)

c[4]=0 because it is initialized as 0 by default when c[0], c[1], c[2] are initialized as 2. (1pt)

11.

Write code which creates a struct type called StudentData to represent the registration data of each student in our class. The structure has two fields: the id field *stu\_id*that is an 32-bit unsigned integer, and the name field *stu\_name* that uses (up to) 20 characters.

struct StudentData {

char stu\_name[32];

unsigned int stu\_id;

} ;

Note:

· unsigned int can be uint32\_t;

· Correct struct definition, including the keyword “struct”, the name (StudentData), brackets ({}) and the semicolon (2pts)

· Correct definition of each field (1pt each)

In the above question, let us assume the struct type in addition has the third field variable *stu\_sec* to represent the student section id, which is of an enum type with three section names SECTION1 (representing 1), SECTION2 (representing 2) and SECTION3(representing 3) as its member fields. Write the code to correctly declare the enum type variable *stu\_sec*.

Ans:

enum {

SECTION1 =1,

SECTION2 =2,

SECTION3 =3

} stu\_sec;

Note:

· Correct enum definition, including the keyword “enum” and brackets ({}. (0.5 pt)

· Correctly create the variable stu\_sec (0.5 pt), NOT to use stu\_sec as the enum type num (enum stu\_sec)

· List SECTION1, SECTION2, and SECTION3 as the member field.

· Correct initialization of the fields. SECTION1 must equal 1 (if SECTION1 is correctly initialized, the other two sections will automatically be initialized as 2 and 3, respectively, so it is okay to not explicitly initialize the latter two. Finally, the different fields are separated by comma, not semi-colon;) (1pt)

Continue with the above question. Write code to (1) declare a variable *bob* and a pointer *p* of this struct type; (2) initialize *p* with the address of *bob;*(3) initialize the section id of *bob*withSECTION3 through the pointer *p*

(1) struct StudentData alice, \*p; (1 pts)

(2) p = &alice; (1 pt)

(3) p->stu\_sec= SECTION2; (2 pts)

(if students use “.” Instead of “->”, take 1 pt off. If it is not stu\_sec = SECTION2, take 2 pts off) (it is okay to assign stu\_sec =2;)