

ITEE



What's New:

The programming questions are designed to help you get in shape for doing assignments.

Learning objectives for this week:

- Understand the major concepts we've covered, including:
 - major computer components
 - major operating system types concepts
 - systems calls and I/O
 - basics of C programming
- Apply these concepts to reasoning about OS components and implementation, including understanding design choices.
- Understand the performance implications, including speed and usability, of major design decisions.

As before, you should aim to answer all of these questions by the end of the course. Being able to answer them by the end of week 3 is a useful goal. Be prepared: if you know what you can't do yourself, you will make most efficient use of the tutorial. The programming aspects are especially important as preparation for the assignments.

1. Operating System Structure

How does a virtual machine operating system differ from a microkernel operating system? Consider the following issues:

- a. Are the two incompatible concepts, i.e., can one have an operating system with features of both?
- b. What are the differences in the interprocess communication models in the two kinds of OS?
- c. Is the Java Virtual Machine any different to the general concept of a virtual machine OS? If so, how does it differ? If not, what are the major points of similarity?

2. Clients and Servers

In each of the following scenarios, which of machines *A–E* is the client and which is the server?

- a. In an X-windows environment, a program is running on machine *A*, displays windows on machine *B*, and a window manager on machine *C* controls the look and feel of the windows.
- b. In the example of part (a), a file is accessed on another computer on the network, machine *D*.
- c. To extend the example further, in the process of accessing the file on another machine on the network, the operating system has to check security information stored on a disk on machine *E*.

3. Mechanism vs. Policy

In a user interface manager, explain which of the following design principles are *mechanism*, and which are *policy*. In each case, comment on whether enforcing the design principle is or is not good for usability.

- a. Ability to draw a window on a different machine to that on which the requesting process is running.
- b. Ability to change the language of text in the interface.
- c. Cut, copy and paste are always done using the same menu items and keyboard commands.
- d. Anything which can be drawn in a window can be printed.
- e. Although it is possible to move the mouse pointer under software control to an arbitrary location, its movement should always be under control of the user, through mouse movement.

4. C programming

Start from the list code given in the lecture notes. Develop extensions as follows, noting which files they should go into (including headers and program files):

- a. Write a loop which visits each node in a list once, without modifying the existing data structures, and prints out the contents (assume all nodes contain a string)
- b. Extend the existing data structures to include a new `struct`, which stores a current location in list as well as a copy of the list head data structure, and can be used to keep track of the next location in the list. Call this new `struct` `Iterator`, and define a type for it using `typedef`.
- c. Define functions which use the new `Iterator` data structure to:
 - i. initialize the iterator with a new list
 - ii. get the next item from its list, moving the current item on (return `NULL` if past the end of the list)
 - iii. report whether there are more items in the list
- d. Use the new type from part (c) to rewrite the loop of part (a) so it uses `Iterator`. Hint: you want code that looks something like this


```
while (is_more (list_loop))
    printf ("next word : %s\n", getNext (&list_loop));
```
- e. Comment on the relative efficiency and ease of use of the two loops you've developed: how does all this compare with an object-oriented language?
- f. Now consider how you would write code to deallocate an entire list. What would you do about deallocating the contents of the list?

5. General: for discussion

Small-scale systems with very small memories are coming back in the form of mobile and embedded devices. What challenges do you think these systems pose for operating systems which are similar to early developments of conventional operating systems (when memories were small, and processors slow)? What do you expect to be different?