CSE 430 Summer 2014 - Homework 2

Ryan Dougherty

Question 3.9

Describe the actions taken by a kernel to context-switch between processes. The OS must save the current process's state (the one that is running) and restore the next-to-be-run process's state. What needs to be saved are all of the registers, the processor id, etc. There are also possible architecture-specific operations that need to be done as well.

Question 3.17

Using the program shown in Figure 3.35, explain what the output will be at lines X and Y. For LINE X, the output will be: "CHILD: 0 CHILD: -1 CHILD: -4 CHILD: -9 CHILD: -16 ". For LINE Y, the output will be: "PARENT: 0 PARENT: 1 PARENT: 2 PARENT: 3 PARENT: 4".

Question 3.18

What are the benefits and the disadvantages of each of the following? Consider both the system level and the programmer level.

(a) Synchronous and asynchronous communication:

- Benefit (synchronous): it allows a rendezvous (when send() and receive() are both blocking) between the sender and receiver.
- Disadvantage (synchronous): a rendezvous may not be required given the context of the system, and the message could have be delievered asynchronously. Therefore, send() and receive() do not have to both block if necessary in some situations.

- Benefit (asynchronous): some scenarios may require asynchronous communication, particularly when there are many users sending messages on the same machine. Also, asynchronous communication can send multiple large messages faster than synchronous (because synchronous blocks until the message send is complete).
- Disadvantage (asynchronous): there is overhead in sending a message asynchronously (in starting a different thread to send the message on, etc.).

(b) Automatic and explicit buffering:

- Benefit (automatic): it provides a queue with an undetermined length. Therefore, the sender does not ever block while it waits to copy some message.
- Disadvantage (automatic): there is no way to figure out how the buffer will be exactly implemented, which may lead to large memory allocation and much of the buffer ends up not being used.
- Benefit (explicit): less likely that memory will be wasted (see automatic's disadvantage above).
- Disadvantage (explicit): since the size of the buffer is known and non-changing, the sender may be blocked while it waits for free space in the buffer.

(c) Send by copy and send by reference:

- Benefit (by reference): there is no extra memory allocation needed to copy the object (unlike by-copy).
- Disadvantage (by reference): it does not allow the receiver to change the input parameter's state.
- Benefit (by copy): it allows the receiver to change the parameter's state (unlike by-reference's disadvantage above).
- Disadvantage (by copy): it requires extra memory allocation and CPU time to copy the contents of the input parameter.

(d) Fixed-sized and variable-sized messages:

- Benefit (fixed-size): a buffer can hold a known number of messages, since both the message and buffer sizes are fixed.

- Disadvantage (fixed-size): only messages of some fixed size can be sent through a buffer.
- Benefit (variable-size): messages of any size can be sent through a buffer (unlike fixed-size's disadvantage above).
- Disadvantage (variable-size): the # of variable-sized messages that can be held by a buffer is unknown, since the size of the message can be anything.

Question 4.8

Which of the following components of program state are shared across threads in a multithreaded process?

(a) Register values: Not shared

(b) Heap memory: Shared

(c) Global variables: Shared

(d) Stack memory: Not shared

Question 4.17

The program shown in Figure 4.16 uses the Pthreads API. What would be the output from the program at LINE C and LINE P? At LINE C, the output is 5, and at LINE P, the output is 0.

Question 4.18

Consider a multicore system and a mulithreaded program written using the many-to-many threading model. Let the number of user-level threads in the program be greater than the number of processing cores in the system. Dicuss the performance implications of the following scenarios.

(a) The number of kernel threads allocated to the program is less than the number of processing cores. In this scenario, some of the processors would remain idle. Only kernel threads are mapped to processors, not user-level threads. Therefore, since the number of kernel threads is less than the number of processing cores, there are some leftover processors, which are idle.

(b) The number of kernel threads allocated to the program is equal to the number of processing cores. In this scenario, it is possible that all of the processors might be utilized at the same time. If a kernel thread blocks for some reason, the processor that is mapped to by that kernel thread will be idle.

(c) The number of kernel threads allocated to the program is greater than the number of processing cores but less than the number of user-level threads. In this scenario, a kernel thread that is blocked (meaning that its corresponding processor is idle) can be swapped out with another that is in the ready state, which will increase usage of the system.