University of Newcastle
School of Electrical Engineering and Computer Science

COMP2240 - Operating Systems
Workshop 2
Topics: Process and Threads

1. Consider an environment in which there is a one-to-one mapping between user-level threads and kernel-level threads that allows one or more threads within a process to issue blocking system calls while other threads continue to run. Explain why this model can make multithreaded programs run faster than their single-threaded counterparts on a uniprocessor computer.

2. Consider the state transition diagram given in Figure 1:

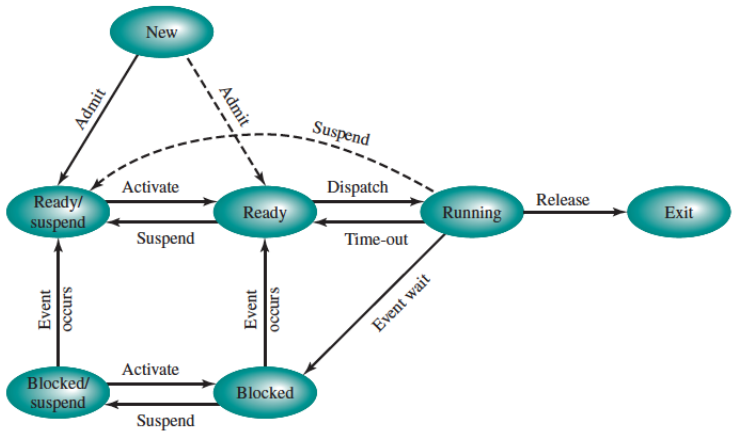


Figure 1: Process State Transition Diagram with Two Suspend States

Suppose that it is time for the OS to dispatch a process and that there are processes in both the Ready state and the Ready/Suspend state, and that at least one process in the Ready/Suspend state has higher scheduling priority than any of the processes in the Ready state.

Two extreme policies are as follows:
(1) Always dispatch from a process in the Ready state, to minimize swapping
(2) Always give preference to the highest-priority process, even though that may mean swapping when swapping is not necessary.

Suggest an intermediate policy that tries to balance the concerns of priority and performance.

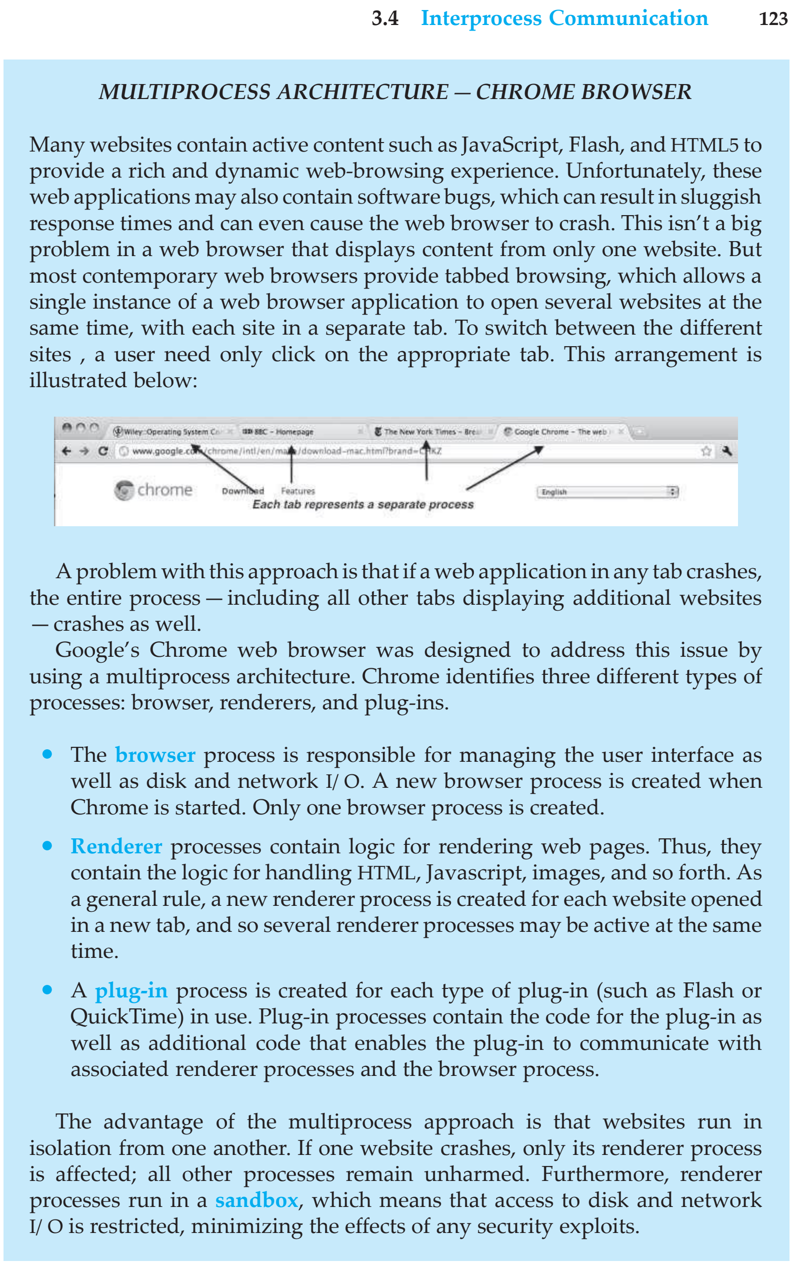
3. Figure 1 in problem 2 suggests that a process can only be in one event queue at a time.

a) Is it possible that you would want to allow a process to wait on more than one event at the same time? Provide an example.
b) In that case, how would you modify the queueing structure of the figure to support this new feature?

4. Consider a multicore system and a multithreaded 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. Discuss the performance implications of the following scenarios.

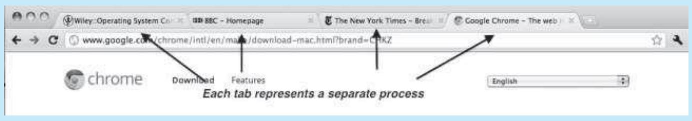
a) The number of kernel threads allocated to the program is less than the number of processing cores.
b) The number of kernel threads allocated to the program is equal to the number of processing cores.
c) The number of kernel threads allocated to the program is greater than the number of processing cores but less than the number of user-level threads.

5. In the given text, it discusses Google's Chrome browser and its practice of opening each new website in a separate process.



MULTIPROCESS ARCHITECTURE — CHROME BROWSER

Many websites contain active content such as JavaScript, Flash, and HTML5 to provide a rich and dynamic web-browsing experience. Unfortunately, these web applications may also contain software bugs, which can result in sluggish response times and can even cause the web browser to crash. This isn't a big problem in a web browser that displays content from only one website. But most contemporary web browsers provide tabbed browsing, which allows a single instance of a web browser application to open several websites at the same time, with each site in a separate tab. To switch between the different sites , a user need only click on the appropriate tab. This arrangement is illustrated below:


Wiley::Operating System Co
BBC – Homepage
The New York Times – Break
Google Chrome – The web
www.google.com/chrome/intl/en/make/download-mac.html?brand=CHKZ
chrome
Download
Features
English
Each tab represents a separate process

A problem with this approach is that if a web application in any tab crashes, the entire process — including all other tabs displaying additional websites — crashes as well.

Google's Chrome web browser was designed to address this issue by using a multiprocess architecture. Chrome identifies three different types of processes: browser, renderers, and plug-ins.

- The **browser** process is responsible for managing the user interface as well as disk and network I/O. A new browser process is created when Chrome is started. Only one browser process is created.
- **Renderer** processes contain logic for rendering web pages. Thus, they contain the logic for handling HTML, Javascript, images, and so forth. As a general rule, a new renderer process is created for each website opened in a new tab, and so several renderer processes may be active at the same time.
- A **plug-in** process is created for each type of plug-in (such as Flash or QuickTime) in use. Plug-in processes contain the code for the plug-in as well as additional code that enables the plug-in to communicate with associated renderer processes and the browser process.

The advantage of the multiprocess approach is that websites run in isolation from one another. If one website crashes, only its renderer process is affected; all other processes remain unharmed. Furthermore, renderer processes run in a **sandbox**, which means that access to disk and network I/O is restricted, minimizing the effects of any security exploits.

Would the same benefits have been achieved if instead Chrome had been designed to open each new website in a separate thread? Explain.

Supplementary problems:

S1. What is a zombie process? What is the importance of a zombie process?

S2. A user's text segment is re-entrant and can therefore be shared. Data and stack segments, on the other hand, are private. What does this mean? Why is it significant?

S3. A system with two dual-core processors has four processors available for scheduling. A CPU-intensive application is running on this system.
All input is performed at program start-up, when a single file must be opened. Similarly, all output is performed just before the program terminates, when the program results must be written to a single file.
Between startup and termination, the program is entirely CPUbound. Your task is to improve the performance of this application by multithreading it. The application runs on a system that uses the one-to-one threading model (each user thread maps to a kernel thread).

a) How many threads will you create to perform the input and output? Explain.
b) How many threads will you create for the CPU-intensive portion of the application? Explain.