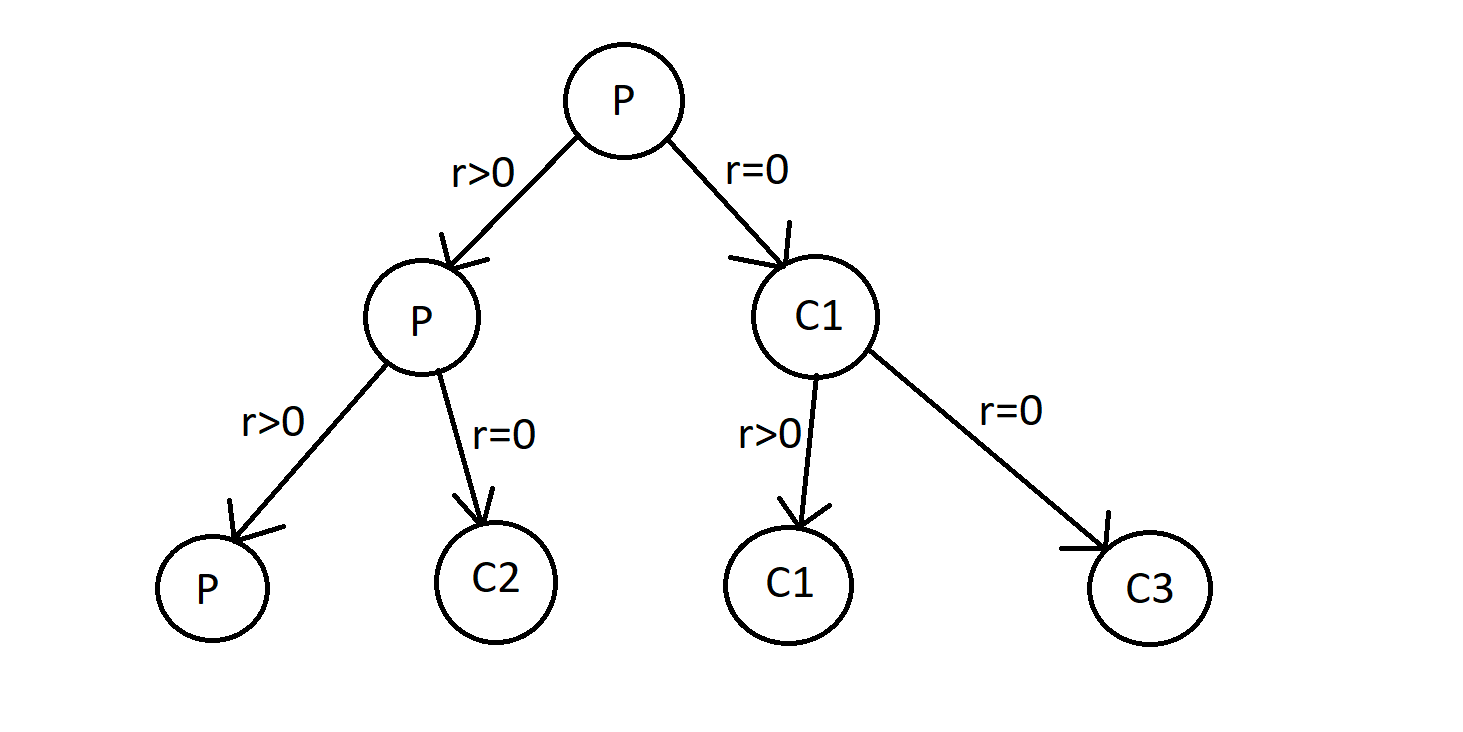
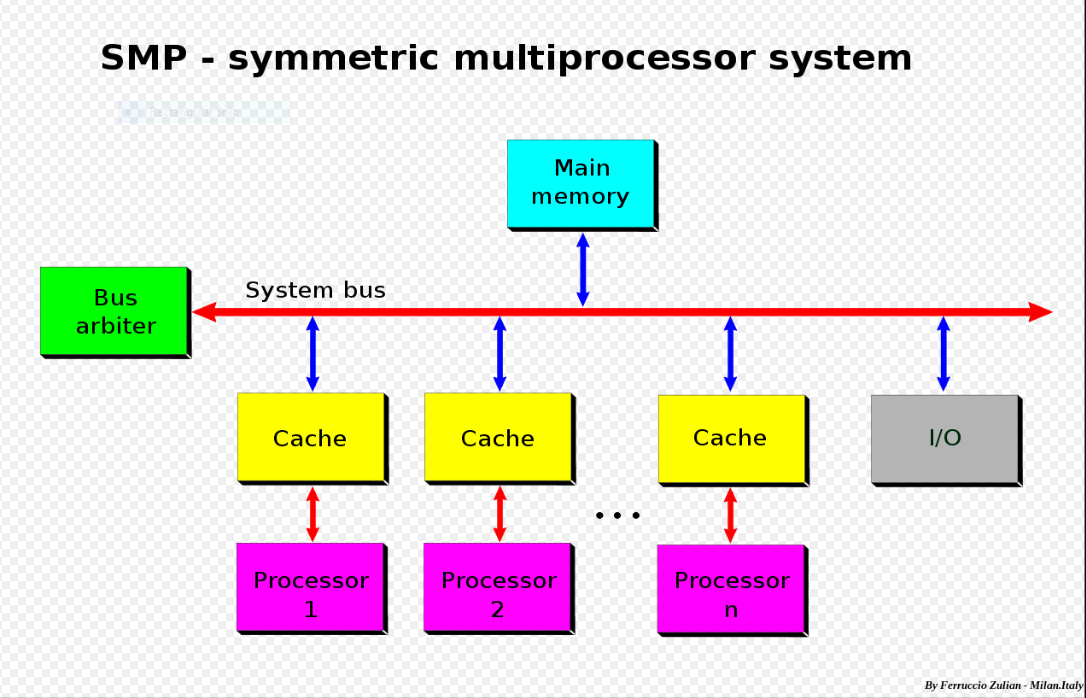
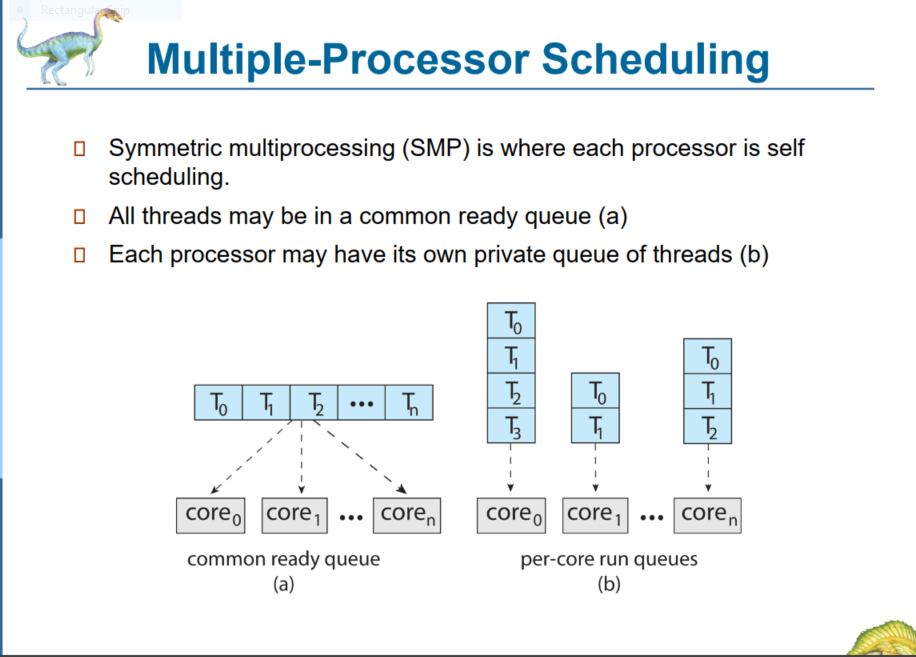
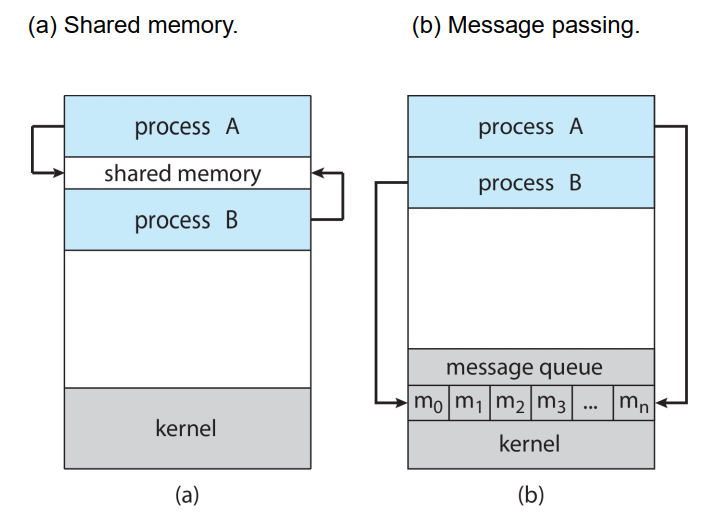
به نام خدا

# فرزان رحمانی - 99521271

# تمرین اول تئوری سیستم عامل

1. اگر در مرحله fork زدن ارور بدهد چون pid<0 است عبارت Fork Failed را چاپ می کند و از برنامه خارج می‌شود. اما اگر در مرحله fork زدن ارور ندهد و با موفقیت انجام شود دو process که یکی child و دیگری parent هست ایجاد می‌شود و هر دو اجرا می‌شوند. زمانی که parent در حال اجرا است pid>0 است و زمانی که که child اجرا می‌شود pid==0. با توجه اینکه در خط 19 از عبارت wait(NULL) استفاده شده است parent صبر می‌کند که یکی از بچه هایش اجرایش تمام شود و چون اینجا فقط یک فرزند دارد لذا child باید اجرا شود و سپس parent اجرا شود.لذا اول child پرینت خود را انجام می‌دهد و سپس parent .  
   خروجی (اگر fork ارور دهد و pid < 0):  
   Fork Failed  
   خروجی (اگر fork ارور ندهد و درست اجرا شود):  
   I'm the child  
   Child Complete
2. در انتها 4 process تولید می شوند. هر یک از این process ها عبارت Who are you?! را 2 بار چاپ می کنند و به همین دلیل در خروجی 8 بار این عبارت چاپ می‌شود. در ابتدا P دو process به اسم P و C1 ایجاد میکند. سپس P دو process P و C2 را بوجود می‌آورد. C1 هم C1 و C3 را ایجاد می‌کند. هر چهار process دو بار عبارت را چاپ می‌کنند.و سپس در خط exit(0) terminate می‌شوند.  
     
     
   خروجی:  
   Who are you?!  
   Who are you?!  
   Who are you?!  
   Who are you?!  
   Who are you?!  
   Who are you?!  
   Who are you?!  
   Who are you?!
3. در این نوع از پردازش هر processor مانند دیگری هست و کار های مشابهی را می‌توانند انجام بدهند.به بیان بهتر:

**Symmetric multiprocessing** or **shared-memory multiprocessing** (**SMP**) involves a multiprocessor computer hardware and software architecture where two or more identical processors are connected to a single, shared main memory, have full access to all input and output devices, and are controlled by a single operating system instance that treats all processors equally, reserving none for special purposes. Most multiprocessor systems today use an SMP architecture.  
  
  
  
  
  
  
and also:  


1.   
    به طور میانگین memory shared سرعت بیشتری دارد. زیرا kernel در روش message passing مجبور به دخالت است و باید پیام ها را لینک کند و ... در حالی که در memory shared بدون واسطه و بدون اتلاف وقت برای ارسال و دریافت پیام ها کار انجام می شود.  
     
    در روش memory shared ، چون باید دیتا ها cache شوند و همین فرآیند ممکن است باعث ایراداتی در به روز بودن دیتا ها شود که گاهی مشکل ساز خواهد بود. اما چنین مشکالتی در passing message نیست زیرا هر process حافظه و data های خودش را دارد و با ارسال پیام ها و دریافتشان تغییرات مربوطه انجام می شود.  
     
   همچنین در ادامه توضیحات بیشتر آمده است:

Message passing and shared memory both have advantages and disadvantages relative to each other. One key dimension for comparing their performance is the amount of time overhead required. In message passing, every piece of data exchanged requires two system calls: one to read and one to write. In addition, the transferred data must be copied twice: once into the kernel memory and once into the receiving process. For a single message, this time penalty is insignificant and is unlikely to affect the process’s performance. However, if the number of messages passed is extremely large, the cumulative effect of this penalty may be significant.  
  
In contrast, shared memory techniques only require a one-time performance penalty during the set-up phase. Once the memory has been shared, there is no additional penalty, regardless of the amount of data transferred. The trade-off is that the penalty for setting up shared memory is significant. The kernel must perform several slow operations to link the shared region to all of the processes’ virtual memory spaces.  
  
Overall, if the two processes will be exchanging a lot of data back and forth repeatedly, shared memory performs very well. While the work to set up the shared memory is expensive, the aggregate effect of the performance is reduced. However, if processes only need to exchange a single message of a few bytes, shared memory would perform very poorly. Message passing techniques impose significantly smaller overhead to set up a one-time data exchange.  
  
Shared memory also has another disadvantage that message passing avoids, which is the problem of synchronization. If both processes try to write to the shared memory region at the same time, the result would be unpredictable and could lead to errors in one or both processes. Consequently, the accesses must be synchronized, meaning that the timing is carefully controlled. Since all message passing exchanges go through the kernel, synchronization techniques are not necessary.  
  
As an example of the synchronization problem, consider an example where the two processes are keeping track of money in your bank account. One process writes a record for a $100 purchase that you made, while another records a $100 deposit. If both are recorded correctly, your new balance should be the same as when the processes started. However, if the account record is in shared memory and the writes are not properly synchronized, the results may not be correct. Instead of your final balance being exactly what you started with, you may end up with $100 more or less than your original balance. Either you or your bank would be very unhappy with this result. Synchronization is a complex topic and we discuss it in its own chapter.

1. بهتر است از kernel-level threads استفاده کنیم.چرا که

A worker thread will block when it has to read a Web page from the disk. If user-level threads are being used, this action will block the entire process, destroying the value of multithreading. Thus, it is essential that kernel threads are used to permit some threads to block without affecting the others.

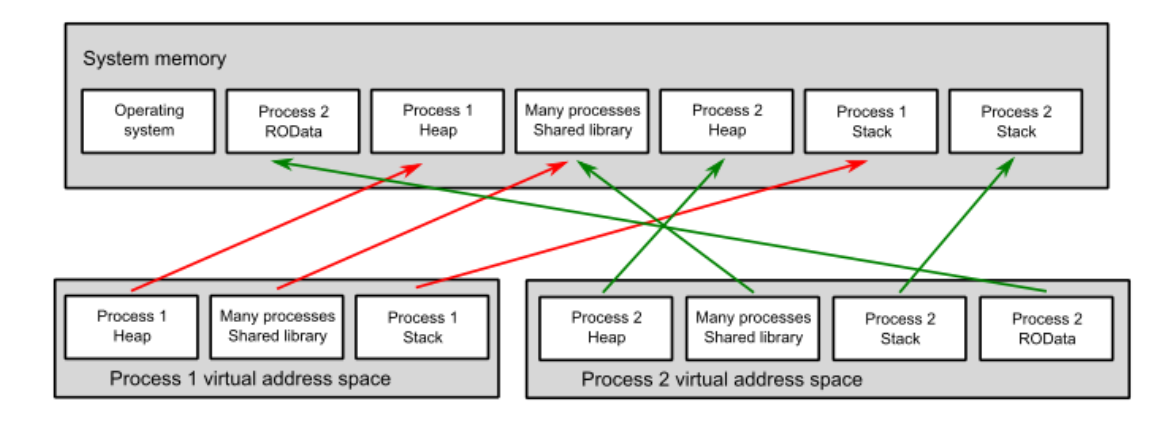
همچنین:  
رشته های سطح kernel معمولا هز ینه (cost) بیشتری نسبت به رشته های سطح کاربر دارند؛ ز یرا توسط داده ساختارهای (data structures) خود kernel ذخیره و نمایش داده میشوند. بطور کلی، برای انجام کارهایی که نیاز به عملیات switch بیشتری بین رشته ها دارند، استفاده از رشته های سطح کاربر توصیه می شود. زیرا سرعت switch بین این رشته ها به دلیل یکجا بودن همه ی آنها در یک پردازه بیشتر است. در مقابل، برای انجام کارهایی که وقفه ها (interrupt) های زمانبری دارند که عموما میتواند بخاطر عملیات O/I باشد و هم چنین برای کارهایی که نیاز به تعداد رشته های بسیار بالا برای انجام دارند، استفاده از رشته های سطح kernel بهتر است .

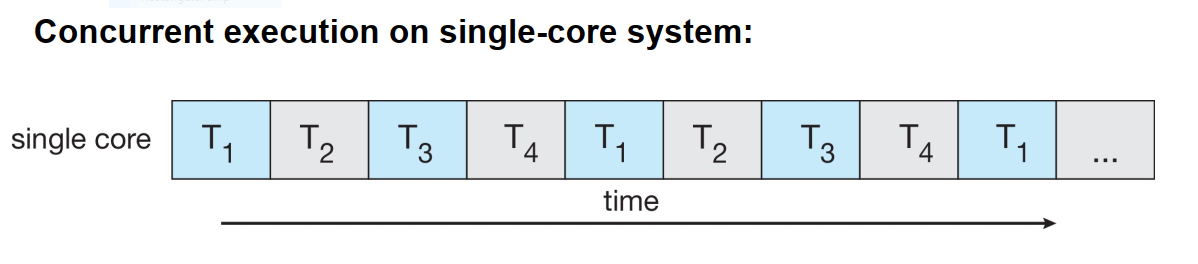
1. ریسمان ها با همکاری یکدیگر می‌خواهند کارهای ما را انجام بدهند و دشمن هم نیستند و چون اینگونه بهتر عملکرد بهتری در کل دارند این کار را انجام می‌دهند.

Threads in a process cooperate. They are not hostile to one another. If yielding is needed for the good of the application, then a thread will yield. After all, it is usually the same programmer who writes the code for all of them.

1. حالت Shared memory segments به اشتراک گذاشته می‌شود.چرا که process جدید نیاز به stack و heap جدید و مستقلی برای خود دارد که از والد کپی می‌کند.

Only the shared memory segments are shared between the  
parent process and the newly forked child process. Copies of the stack and the heap are made for the newly created process.

  
In the shared-memory model, a region of memory that is shared by the cooperating processes is established. Processes can then exchange information by reading and writing data to the shared region.  
A shared-memory region resides in the address space of the process creating the shared-memory segment. Other processes that wish to communicate using this shared-memory segment must attach it to their address space. Normally, the operating system tries to prevent one process from accessing another process’s memory (viruses try to break this). Shared memory requires that two or more processes agree to remove this restriction. They can then exchange information by reading and writing data in the shared areas. The form of the data and **the location are determined by these processes** and are not under the operating system’s control. The processes are also responsible for ensuring that they are not writing to the same location simultaneously.

1. Speedup کل سیستم تقریبا برابر با 1.292 می باشد. تقریبا 29% speedup داریم.  
   Speedup =   
     
   Speedup =
2.   
   • مدیریت دسترسی پردازه های مختلف اگر بخواهند به داده های مشترکی دسترسی داشته باشند، که هر پردازه به چه بخشی دسترسی داشته باشد یا اگر تغییری توسط یکی انجام شد در بقیه آپدیت شود و ... از چالشهای پردازه های همزمان است.  
    • برقراری ارتباط بین پردازه های همزمان توسط سیستم عامل  
   • مشخص کردن بخش هایی از برنامه که میتوانند همزمان انجام شوند و تفکیک آن ها از بخش هایی که باید به صورت متوالی انجام شوند. همچنین پیاده سازی برنامه ها به صورتی که حداکثر استفاده از همزمانی را ببریم نیز یکی از چالش هاست.

a. A method of time sharing must be implemented to allow each of several processes to have access to the system. This method involves the preemption of processes that do not voluntarily give up the CPU (by using a system call, for instance) and the kernel being reentrant (so more than one process may be executing kernel code concurrently).   
  
b. Processes and system resources must have protections and must be protected from each other. Any given process must be limited in the amount of memory it can use and the operations it can perform on devices like disks.   
  
c. Care must be taken in the kernel to prevent deadlocks between processes, so processes aren’t waiting for each other’s allocated resources.  
  
  
d. The CPU scheduler must be aware of the different concurrent processes and must choose an appropriate algorithm that schedules the concurrent processes.   
  
e. Concurrent processes may need to communicate with one another, and the operating system must therefore develop one or more methods for providing interprocess communication.   
  
f. The operating system must manage memory to support multiple concurrent processes.

# پایان