بسم الله الرحمن الرحیم

محمد عرفان زارع زردینی

تمرین تئوری سری اول

(1

(در متن پاسخ(فرایند=process))

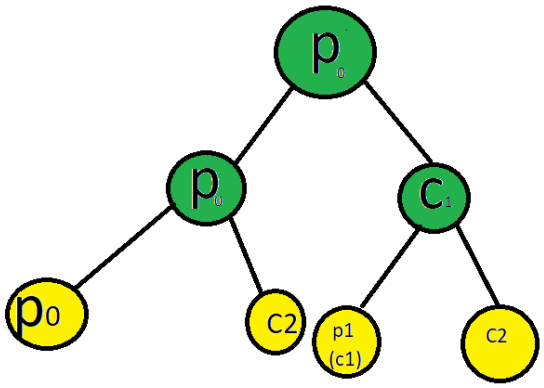
در خط شماره 7 ، با استفاده از fork ، processجدیدی ایجاد می شود و process id آن ذخیره می شود . در شرط if چک می شود که اگر pid کوچک تر از صفر باشد ، یعنی fork failed رخ داده که با stderr(مانند stdoutهست و معمولا برای چاپ خطا استفاده می شود. و return 1 هم بدین معنا هست که در اجرای برنامه ،خطایی وجود دارد و آنچه قرار بر انجام است(عمل ایجاد فرایند جدید) را درست انجام نمی دهد . خط 14 چک می کند که آیا process اجرا کننده ، فرزند هست یا خیر. در صورتی که process فرزند بود، متن خط 15 چاپ می شود که این با توجه به اینکه فرایند فرزند هم در حال اجرا برنامه هست(می دانیم گه هردو فرایند همزمان در حال اجرا کد ها هستند تا اینجای کار) توسط این process اجرا می شود .زمانی وارد else می شود که process در حال اجرا، والد باشد که در خط19 دستور نوشته شده بدین معناست که تا زمانی که فرایند فرزند، killنشده فرایند پدر صبر میکنه. سپس وقتی فرزند کارش تمام شد و kill شد،فرایند پدر متن خط20 را چاپ می کند و بدین معناست که کار فرایند فرزند تمام شده.خروجی هم به شکل زیر هست:

```
I'm the child
Child Complete
```

2)

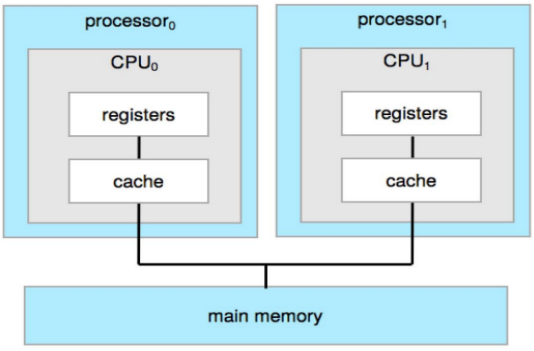
فرایند=Process

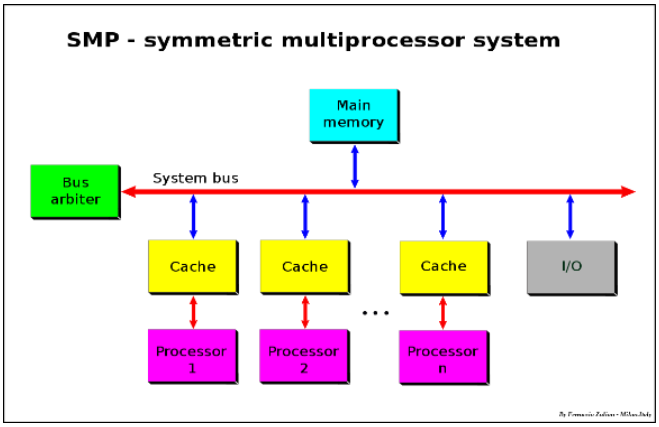
در تابع main ، ابتدا تابع نوشته شده را فراخوانی می کنیم. در تابع مدنظر ابتدا فرایند اصلی(p0) به دو فرایند اصلی و فرزند اول(c0) تبدیل می شود (سطر 1) سپس با دستور fork دوم، دوتا فرایند موجود هرکدوم به دو فرایند(خود فرایند و فرزند آن تبدیل می شود . طبق شکل زیر می توان مفهوم این توضیحات رو بیان کرد که بدان tree process گویند. سپس خط بعد تابع که متن مدنظر را چاپ می کند توسط هر 4 فرایند انجام می شود و تابع پایان می پذیرد. حال تابع main ادامه می یابد و متن مد نظر توسط هر 4 فرایند چاپ می شود. زیرا هنوز هیچ کدام از فرایند ها kill نشده اند. پیرامون دستور exit() می آید فرایند مدنظر را kill می کند .بدین صورت برنامه اجرا می شود و 8 بار جمله مدنظر در کل چاپ می شود.



3)

در ساختار مورد بحث ؛ همه پردازنده ها یکسان هستند و میتوانند تمامی کارها را انجام دهند .هر cpu دارای ریجستر ها و کش مربوط به خود (مستقل) هستند و main memoryمیان process ها ارتباط برقرار می نماید. بخش main memory قسمتی از مرحله sync کردن را انجام می دهد. Sync کردن در واقع در نحوه خوندن واتصالات انجام می شود. همچنین cache دیتا را از main memory می خواند وcache می کنند.





4)

Massege passing : به فرایند ها اجازه می دهد تا داده ها رو بدون اتصال به یکدیگر بخواند و در صف پیام قرار دهد. پیام ها تا زمانی که گیرنده آن ها را بازیابی کند در صف ذخیره می شوند. صف های پیام برای ارتباط بین process ها مفید و کاربردی هست.از دیگر مزیت های این روش ،این است که ساخت سخت افزار موازی آسانتر هست. این به این دلیل است که این روش نسبت به تاخیر های ارتباطی بالاتر قابلیت تحمل و هندل کردن آن را دارد. همچنین پیاده سازی آن نسبت به shared memory mdoel ،آسانتر است. این در حالیسست که که این روش نسبت به shared memory ، communication کندتری دارد.به این دلیل که در این روش تنظیم اتصال به زمان نیاز دارد. در این روش worker ها از طریق massege passsing با هم در ارتباط اند و پیام ها رو جدا نگه می دارند بگونه ای که workers نمی توانند داده های هم را تغییر دهند.

Shared memory:

حافظه مشترک در این روش، حافظه ای است که می تواند به طور همزمان توسط چندین process به آن دسترسی داشت. این کار به گونه ای انجام می شود که process بتوانند با یکدیگر ارتباط برقرار کنند. مزیت مدل حافظه مشترک این است که ارتباط حافظه در مقایسه با مدل massege passing در دستگاه یکسان سریعتر است.

این درحالی است که مدل حافظه مشترک ممکن است مشکلاتی مانند همگام سازی و محافظت از حافظه ایجاد کند که باید برطرف شود.این روش برای برنامه هایی با برنامه نویس های موازی کاربرد دارد.همچنین در این روش چند worker برروی یک دیتا کار می کنند.

Massage passing معمولا در سیستم توزیع شده استفاده می شود.

Shared memory معمولا در سیستم های تک (single) استفاده می شود.

5)

زمانی که یک worker از thread ؛ یک web page را از دیسک می خواند؛ مسدود می شود. هنگامی که از thread های user-level استفاده شود ؛ سبب می شود کل processرا مسدود می کند و سبب کاهش value بخش thread می شود. بنابر این باید از kernel thread برای permitکردن تعدادی threadدر جهت block کردن بدون تاثیر بر بقیه استفاده شود.

6)

یک thread با فراخوانی thread yield به طور داوطلبانه خود را به cpu ارائه می دهد. دلایل آن به شرح زیر می‌باشد:

ا–برنامه اشاره شده درقسمت قبل؛در یک process کد ها رو برای تمام thread ها می نویسد.

۲- thread در یک فرایند با هم همکاری و اگر برای بهبود برنامه نیاز به yielding بود، آنگاه thread به طور قطع yield خواهد شد.

۳-برای پیشگیری از deadlock ؛ گاهی لازم است که یک thread خود را yield کند تا فرآیند به درستی ادامه یابد.

۴-همچنین thread schedulerمانند سیستم عامل نداریم.از این رو نمیخواهیم که یک thread ،به حالت بی حرکت درآید و همیشه cpu time فرایند ها را بگیرد.

۷)

بخش shared memory تنها میان فرایند والد و فرایند فرزند حاصل شده از fork به اشتراک گذاشته می شود و از آن برای تبادل داده ها استفاده می شود .کپی هایی از stack و heap برای فرایند جدید ایجاد شده ؛ ساخته می شود.چون دستور fork در واقع میاید یک process جدید ایجاد می کنه ودر نتیجه فرایند جدید نیاز به stack و heap دارد. به همین دلیل از والدش کپی کرده و برایش در نظر می گیرد. از این به بعدheap وstack شان مستقل می‌شود.

۸)

$S = ((1 - fE) + (fE / fI))^{-1}$

طبق فرمول بالا و با اطلاع از این که $f_E=0.4$ هست و $f_I=2.3$ ؛ با جایگذاری در فرمول داریم:

$1/((1-0.4)+(0.4/2.3)) = 1.29213483146 = 1.29$

در نتیجه مقدار speddup برابر است با ۱.۲۹

۹)

۱–برای پیشگیری از deadlock بین process ؛ باید از kernel محافظت کرد، پس برای همین processها منتظر منابع تخصیص یافته یکدیگر نباید بمانند.

۲– processها و system resource باید دارای محافظی باشند و از یکدیگر محافظت کنند. هر process داده شده(مورد بررسی) باید از نظر مقدار حافظه ای که می تواند استفاده کند و operation هایی که می تواند روی دستگاه هایی مانند دیسک انجام بدهد محدود باشد .

۳–متد time sharing باید اجرا شود تا به هر یک از چند process اجازه دسترسی به سیستم را بدهد . این متد شامل پیشگیری processهایی هست که به‌طور داوطلبانه CPU را رها نمی‌کنند (مثال با استفاده از یکsystem call) و kernel دوباره وارد می‌شود (پس ممکن است بیش از یک فرآیند همزمان کد kernel را اجرا کنند).