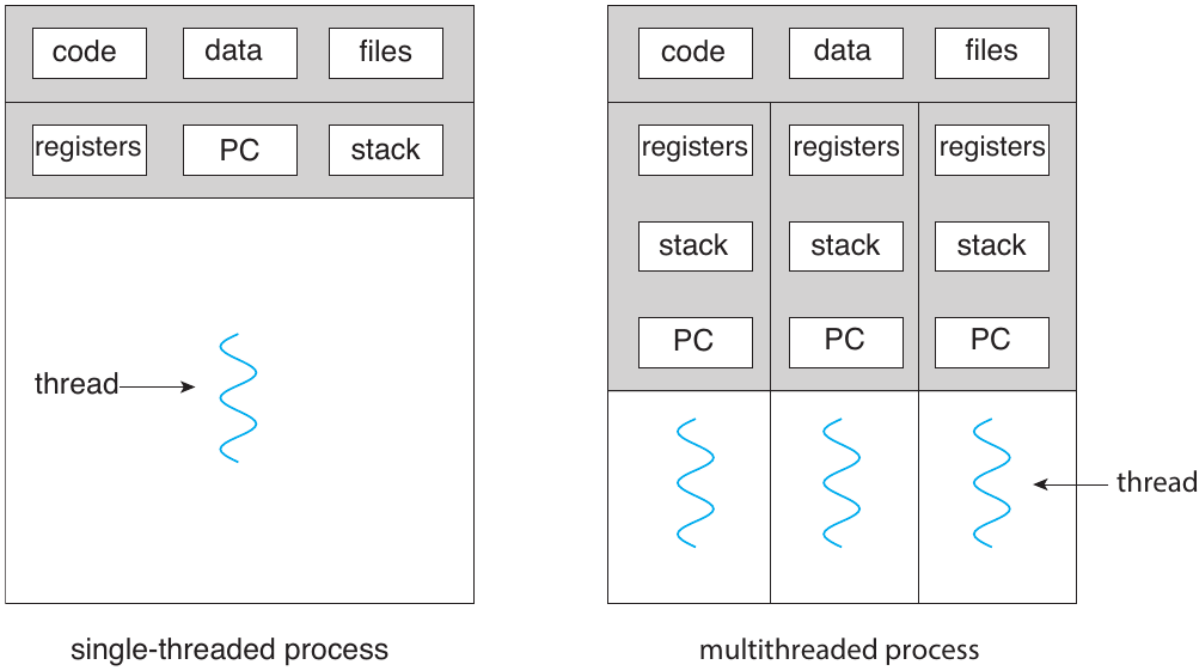
پاسخ تکلیف دوم سیستم عامل

پاییز ۹۸

دانشگاه صنعتی اصفهان

سوال ۱:

الف)



single-threaded process

multithreaded process

با توجه به شکل بالا، یک فرآیند تک نخی دارای یک کد و داده و فایل است و همچنین تنها یک فضای حافظه از استک دارد و علاوه بر آن از یک دسته ریجستر استفاده می‌کند و تنها یک ریجستر program counter دارد. اما در یک فرآیند چند نخی ممکن است بعضی از از این منابع به تعداد نخ‌ها بیشتر شوند؛ برای مثال همان گونه که در تصویر قابل مشاهده است: در یک فرآیند چند نخی، تمام نخ‌ها از یک مجموعه داده، فایل و کد بهره می‌برند اما ریجسترهای مورد استفاده و فضای گرفته شده از استک و همچنین program counter افزایش پیدا می‌کند.

ب) در عملیات تعویض متن، اطلاعات فرآیندی که قرار است از پردازنده خارج شود در PCB مربوط به فرآیند ذخیره می‌شود و اطلاعات فرآیندی که قرار است برای پردازش وارد پردازنده شود خوانده و تنظیم می‌شود. در واقع PCB، اطلاعات فرآیندها را در خود ذخیره می‌کند تا هنگام عملیات تعویض متن، اطلاعات و وضعیت آن فرآیند از دست نرود.

در سیستم لینوکس برای هر thread یک ساختمان داده به نام task_struct در فایل linux/sched.hقرار دارد که حاوی اطلاعات PCB است.

ج) تفاوتها: سیگنال یک ایونت است که توسط cpu ایجاد میشود و به یک پروسس ارسال می‌گردد، در صورتی که اینتراپت سخت افزاری توسط یک دیوایس خارجی ایجاد و به cpu ارسال می‌شود. گاهی هم یک پروسس که در حال اجرا در cpu است سیگنال را ایجاد می‌کند (مانند اینتراپت نرم‌افزاری).

سیگنال مبتنی بر سیستم عامل است لذا سیستم عامل‌های مختلف سیگنال‌های متفاوتی خواهند داشت. اما اینتراپت سخت‌افزاری وابسته به سخت‌افزار مربوطه است.

شباهت‌ها: با رخداد هردو، رویکرد کلی، رسیدگی سریع به آنها است و هر دو قابل mask هستند. وقتی سیگنال دریافت میشود یک تابع به نام هندلر سیگنال اجرا می‌شود و وقتی اینتراپت دریافت می‌شود یک سرویس روتین حاوی کد مربوط به اینتراپت اجرا می‌شود.

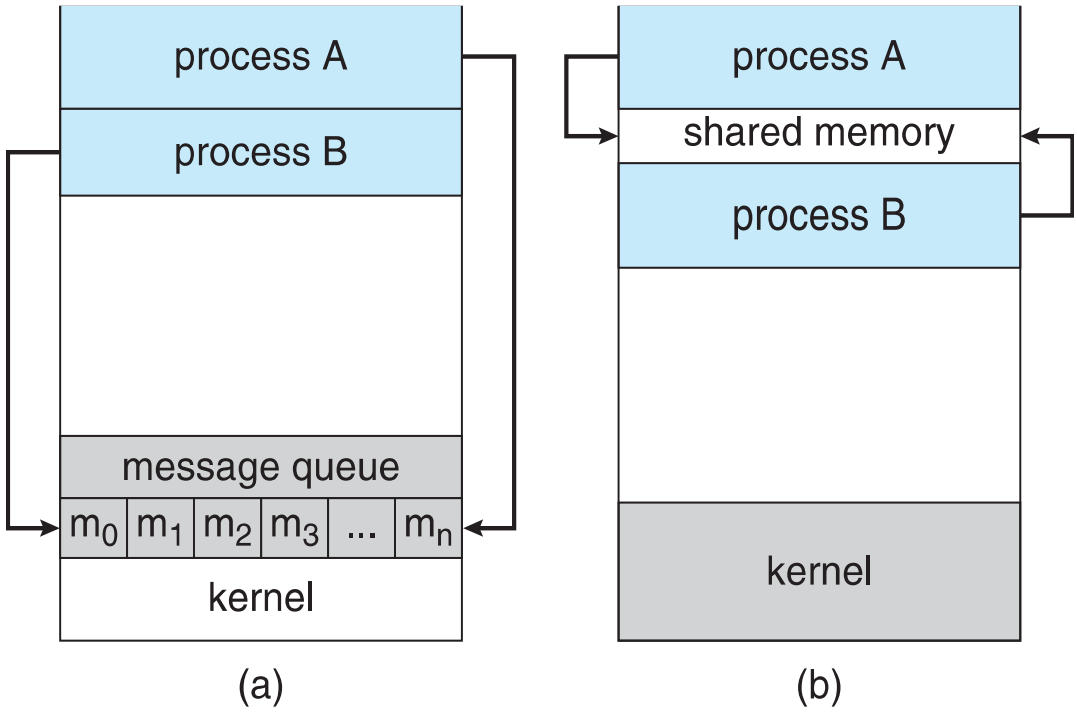
د) ۱. Scheduler فعال شده و پروسس موردنظر که در حالت آماده است را برای اجرا انتخاب کرده است.

۲. اگر فرآیند به I/O نیاز داشته باشد یا یک فراخوانی همانند wait یا sleep اجرا کند.

۳. اگر رم برای اجرای پروسس‌ها کم بیاید و سیستم مجبور شود یک پروسس که در حالت انتظار است را suspend کند تا پروسس‌های دیگر بتوانند اجرا شوند.

جواب سوال ۲:

الف) این دو مدل عبارتند از ۱. حافظه مشترک ۲. انتقال پیام.



در مدل حافظه ی مشترک یک حافظه مشترک بین دو فرآیند ایجاد شده که جزو فضای حافظه هر دو فرآیند محسوب می‌شود و از این رو هر دو فرآیند به آن دسترسی دارند. سپس دو فرآیند داده های خود را بر روی این حافظه قرار می‌دهند و از این طریق با هم ارتباط برقرار می‌کنند. اما در مدل انتقال پیام، پروسس ها برای هم پیام میفرستند و یک پروسس فرستنده پیام و دیگری گیرنده پیام میشود در این حالت پروسس ابتدا از طریق یک فراخوانی سیستمی، پیام خود را به سیستم عامل می‌دهد و سپس سیستم عامل آن را به فرآیند مقصد تحویل می‌دهد(massage Queue ساختاری است که در کرنل سیستم عامل پیاده شده است و از آن برای "نگهداری پیام‌ها تا هنگام تحویل" استفاده می‌شود تا هیچ پیامی از دست نرود).

مزیت استفاده از shm سرعت انتقال اطلاعات است در حالی که عیب آن سختی کنترل همزمانی و تولید و مصرف اطلاعات در حافظه مشترک است و این کار را برای برنامه نویس سخت می‌کند. اما message passing سرعت پایین‌تری دارد چون سربار مدیریت آن

در سیستم عامل نسبت به حافظه مشترک زیاد است اما مزیت آن راحتی استفاده از آن است زیرا ارسال پیام توسط لایه‌های سیستم عامل پیاده‌سازی شده است.

ب) سرعت حافظه‌ی مشترک نسبت به روش انتقال پیام بسیار بیشتر است به گونه‌ای که می‌توان گفت امروزه سریع‌ترین روش برای ارتباط بین دو فرآیند محسوب می‌شود.

ایراد حافظه‌ی مشترک نسبت به روش انتقال پیام، این است که کنترل و هماهنگ شدن دو فرآیند مقصد و مبدا بر عهده‌ی خود برنامه نویس است(نباید همزمان از حافظه‌ی مشترک خوانده و یا بر روی آن نوشته شود) و سیستم عامل هیچ گونه کنترلی نمی‌کند بلکه تنها امکان پیاده‌سازی این روش را فراهم می‌آورد، اما در روش انتقال پیام، از آنجا که پیام‌ها به کرنل سیستم عامل تحویل داده می‌شود، سیستم عامل می‌تواند هماهنگی مورد نیاز را بین فرآیندها ایجاد کند.

سوال ۳:

مقدار متغیر var در والد و فرزند متفاوت است که این نشان می‌دهد حافظه var در دو پروسس متفاوت است و مشترک نیست. از طرف دیگر مقدار آدرس var در هر دو پروسس یکسان است در صورتی که انتظار داریم آدرس var در حافظه فیزیکی در دو پروسس متفاوت باشد چون حافظه مشترک ندارند. در واقع این مسیله نشان می‌دهد که آدرسی که میبینیم آدرس واقعی فیزیکی نیست بلکه یک آدرس مجازی است که محل قرارگرفتن این متغیر به نسبت ابتدای فضای حافظه هر پروسس را نشان می‌دهد و چون فضای آدرس دو پروسس تا قبل از فراخوانی fork برای هر دو پروسس یکسان است آدرس مجازی یا نسبی در هردو یکسان است (در فصل ۹ و ۱۰ کتاب حافظه مجازی را میخوانید)

سوال ۴:

برای مورد اول، چون ممکن است دو فرآیند بر روی یک سیستم اجرا نشوند و بلکه از طریق شبکه به هم متصل گردند بنابراین استفاده از روش‌هایی همانند حافظه مشترک، خط لوله

معمولی و خط لوله‌ی نامگذاری شده نمی‌توان استفاده کرد. بنابراین دو روش سوکت و RPC باقی می‌ماند که از لحاظ سرعت برقراری ارتباط تفاوت چندانی با هم ندارند.

اما از آن جا که این عملیات یک عملیات پیچیده است و نیازمند تعریف پروتکل خاصی بین دو فرآیند است تا استفاده از آن توسط برنامه نویس راحت‌تر گردد، بهتر است از RPC استفاده شود.

برای مورد دوم، از آنجا که رابطه‌ی پدری-فرزندی بین دو فرآیند برقرار است بنابراین دو فرآیند بر روی یک سیستم قرار دارند. بنابراین بهتر است از روشی استفاده کنیم که بیشترین سرعت و سادگی را فراهم کند.

پس در این حالت بهتر است از خط لوله‌ی معمولی استفاده کنیم.

سوال ۵:

الف) چنانچه از روش thread pool استفاده کنیم قسمتی از سربار ایجاد نخها از آنلاین به آفلاین منتقل می‌شود. زیرا نخ‌ها قبلا ایجاد شده‌اند و تنها منتظر هستند تا یک کار به آن‌ها نظیر شود. همچنین در این روش می‌توان منابع سرور را نیز کنترل کرد زیرا در thread pool تنها می‌توان تعداد معینی نخ ایجاد کرد اما اگر به صورت خالص از thread استفاده کنیم با توجه به اینکه درخواست‌هایی که به سرور آمازون ارسال می‌شود بسیار زیاد است احتمال پرشدن تمام نخهای ممکن سیستم و از دست رفتن بعضی درخواست‌های کاربران وجود دارد. در صورتی که در thread pool در صورتی که تعداد نخ‌های فعال به حداکثر ظرفیت pool برسد کتابخانه thread pool درخواست‌های جدید برای ساخت نخ را در صفی قرار می‌دهد تا به محض آزادشدن یک نخ، درخواست از صف خارج و به نخ مربوطه assign شود

ب) با اینکه thread pool دارای صف است ممکن است صف ایجاد شده هم پر شود. در آمازون هم همین اتفاق افتاد و درخواست‌ها آنقدر زیاد شد که حتی صف pool هم پر شد و بنابراین درخواست‌های بعدی از دست رفتند.

جواب سوال ۶:

الف) با توجه به کامنت‌ها، تابع f برای ضرب سطرهای i تا j یک ماتریس در ستون‌های k تا l ماتریس دیگر استفاده می‌شود. بنابراین برای محاسبه نیاز به i,j, k, l دارد. برای ارسال این مقادیر به تابع f از یک structure استفاده می‌کنیم.

```
Struct row_col{
        int r1;
        int r2;
        int c1;
        int c2;
}
int m1[100];
int m2[100];
int m_reply[100];

f1(void*);
read_matrix(char *matrix_name);
print_matrix(int *m);

int main(int argc, char* argv[]){
```

```
	pthread_t t1, t2;
	read_matrix(argv[1]);
	read_matrix(argv[2]);
	struct row_col input1;
	input1.r1=0; input1.r2=49; input1.c1=0;input1.c2=49;

	pthread_create(t1, NULL, f, (void*)input1);
	struct row_col input2;
	input2.r1=50; input2.r2=99; input2.c1=50;input2.c2=99;

	pthread_create(t2, NULL, f, (void*)input2);
	pthread_join(t1, NULL);
	pthread_join(t2, NULL);
	print_matrix(m_reply);
}
```

برنامه‌ی p2.c:

```
int main(){
	pid_t pid;
	pid = fork();
	if(pid == 0){
		execl(“./p1”, “m1.txt”, “m2.txt”);
	}else{
		wait(NULL);
```

```
        pid = fork();
        if(pid == 0){
            execl(“./p1”, “mr.txt”, “m3.txt”);
        }else{
            wait();
            Show_m(“mr.txt”);
        }
    }
```

ب) در برنامه‌ی P1 از موازی‌سازی داده استفاده شده است زیرا تنها یک عمل(ضرب ماتریسی) بر روی بازه‌ی متفاوتی از داده‌ها توسط دو نخ انجام می‌شود. اما در برنامه p2 نتوانستیم موازی سازی داشته باشیم زیرا باید منتظر اتمام محاسبه ضرب دو ماتریس اول باشیم تا بتوانیم خروجی آن را در ماتریس سوم ضرب کنیم.

ج) ذخیره‌ی بر روی دسیک، عملیات I/O محسوب می‌شود و وقت زیادی را تلف می‌کند. اگر برنامه‌ی P1 به جای ذخیره‌ی حاصل ضرب دو ماتریس اول در یک فایل بر روی دیسک، آن را از طریق خط لوله به P2 منتقل می‌کرد سرعت اجرای عملیات بیشتر می‌شد همچنین از طرفی P2 می‌توانست به محض آماده شدن سطر اول حاصل از ضرب دو ماتریس اول، عملیات ضرب ماتریس سوم را آغاز کند و منتظر کامل شدن نتیجه‌ی ضرب دو ماتریس قبلی نماند. یعنی در p2 نیز می‌توانستیم از طریق multiprocessing موازی‌سازی داده داشته باشیم.