دانشگاه صنعتی امیرکبیر

آزمایشگاه سیستم‌عامل

جلسه چهارم: نحوه برقراري ارتباط بين دو پردازه

مدرس :مینا یوسف‌نژاد

در سیستم‌عامل‌ها، فرآیندها برنامه‌هایی هستند که در حال اجرا هستند و می‌توانند به طور همزمان در حافظه فعال باشند. هر فرآیند می‌تواند مستقل یا همکار باشد:

- **فرآیندهای مستقل**

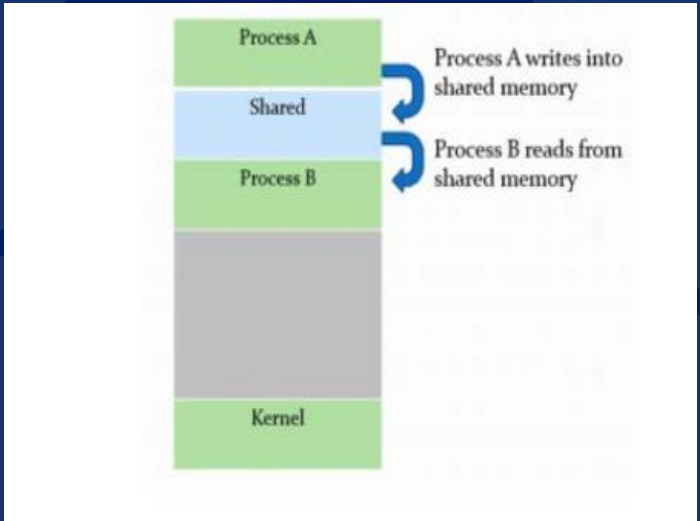
فرآیندهای مستقل هیچگونه تعامل و وابستگی به سایر فرآیندها ندارند. این فرآیندها بدون تأثیرپذیری از دیگران اجرا می‌شوند و داده‌هایشان را با سایر فرآیندها به اشتراک نمی‌گذارند. در نتیجه، اجرای مستقل آنها تضمین می‌کند که تغییرات در این فرآیندها تأثیری بر سایرین نخواهد داشت.

- **فرآیندهای همکار**

فرآیندهای همکار می‌توانند بر یکدیگر تأثیر بگذارند و اغلب برای انجام کارهای مشترک با یکدیگر ارتباط برقرار می‌کنند. این فرآیندها به صورت همزمان بر داده‌ها تأثیر می‌گذارند و با به اشتراک گذاشتن اطلاعات، سرعت و کارایی را افزایش می‌دهند.

ارتباطات بین فرآیندی (IPC - Interprocess Communication) ارتباطات بین فرآیندی سازوکاری است که به فرآیندها اجازه می‌دهد تا برای تبادل داده‌ها، همگام‌سازی و بهینه‌سازی عملکرد با یکدیگر تعامل کنند. دو رویکرد اصلی برای ارتباطات بین فرآیندی وجود دارد:

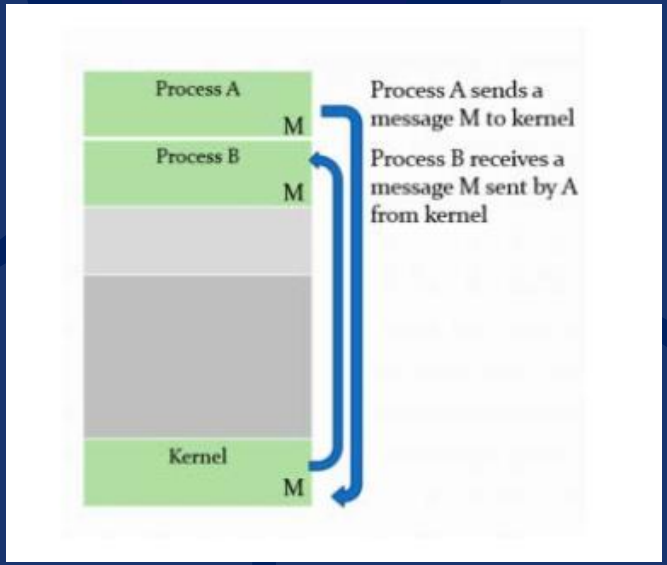
- حافظه مشترک (Shared Memory): در این رویکرد، بخشی از حافظه به صورت مشترک بین فرآیندها قابل دسترس است. این فرآیندها می‌توانند داده‌ها را مستقیماً از این حافظه بخوانند یا در آن بنویسند. استفاده از حافظه مشترک سریع و کارآمد است زیرا تبادل داده‌ها مستقیماً از حافظه سیستم انجام می‌شود. با این حال، نیازمند همگام‌سازی است تا تداخل‌های احتمالی (مانند همزمان نوشتن یا خواندن نادرست) جلوگیری شود.
- ارسال پیام (Message Passing): در این رویکرد، فرآیندها از طریق ارسال و دریافت پیام‌ها با یکدیگر ارتباط برقرار می‌کنند. پیام‌ها به صورت ساختارهای داده‌ای (مانند صف‌های پیام) بین فرآیندها جابجا می‌شوند. این روش از لحاظ همگام‌سازی به نسبت ساده‌تر از حافظه مشترک است و به فرآیندها اجازه می‌دهد بدون نیاز به حافظه مشترک، داده‌ها را تبادل کنند. این روش معمولاً برای ارتباط بین فرآیندهای جداگانه، مثل فرآیندهای در شبکه‌های مختلف یا سیستم‌های توزیع شده، مناسب است.



فرآیندهای همکار Process A و Process B از طریق یک بخش حافظه مشترک (Shared Memory) با یکدیگر ارتباط برقرار می‌کنند.

در این ساختار:

- Process A می‌تواند داده‌ها را در قسمت حافظه مشترک بنویسد.
- Process B می‌تواند از همان قسمت حافظه مشترک داده‌ها را بخواند.



فرآیندهای A و B از طریق روش ارسال پیام (Message Passing) با یکدیگر ارتباط برقرار می‌کنند.

- Process A یک پیام به نام M را به کرنل (Kernel)ارسال می‌کند.
- Kernel پیام را پردازش کرده و سپس آن را به Process Bارسال می‌کند تا دریافت کند.

Shmget

این تابع برای ایجاد یک بخش حافظه مشترک جدید یا دسترسی به یک بخش حافظه موجود استفاده می‌شود.

```
int shmget(key_t key, size_t size, int shmflg);
```

- **key:** برای ایجاد یک قطعه حافظه جدید، مقدار IPC_PRIVATE به آن داده می‌شود، که در این صورت یک شناسه جدید برمی‌گرداند که توسط سایر فرآیندها قابل استفاده است.
- پارامتر size: اندازه حافظه مورد نیاز را مشخص می‌کند. اگر قطعه از قبل وجود داشته باشد، اندازه نباید از اندازه اولیه بزرگتر باشد.
- **shmflg**: برای تنظیم مجوزهای دسترسی و گزینه‌های خاص استفاده می‌شود. به عنوان مثال، S_IRUSR | S_IWUSR دسترسی خواندن و نوشتن به کاربر می‌دهد.

در صورت موفقیت، این تابع یک مقدار integer به عنوان شناسه (ID) برای حافظه مشترک برمی‌گرداند.

shmat

این تابع برای اتصال یک بخش حافظه مشترک به فضای آدرس فرآیندی که آن را فراخوانی می‌کند استفاده می‌شود، که به این فرآیند امکان دسترسی به حافظه مشترک را می‌دهد.

```
void *shmat(int id, const void *addr, int flags);
```

- **id**: شناسه حافظه مشترک که توسط shmget برگردانده شده است.
- **addr**: آدرس مورد نظر برای اتصال به حافظه مشترک. اگر مقدار آن NULL باشد، سیستم به صورت خودکار یک آدرس انتخاب می‌کند.
- **flags**: برای مشخص کردن گزینه‌های دسترسی به حافظه مشترک است؛ برای ساده‌سازی معمولاً مقدار ۰ به آن داده می‌شود.

در صورت موفقیت، این تابع یک اشاره‌گر به محل اتصال حافظه مشترک برمی‌گرداند.

shmdt

این تابع برای جدا کردن یک بخش حافظه مشترک از فضای آدرس فرآیندی که به آن متصل است استفاده می‌شود.

```
int shmdt(const void *addr);
```

- **addr**: آدرس اشاره‌گر به قطعه حافظه مشترکی که باید از فضای آدرس جدا شود.

این تابع در صورت موفقیت مقدار ۰ برمی‌گرداند.

shmctl

این تابع برای انجام عملیات کنترلی روی بخش حافظه مشترک استفاده می‌شود. با استفاده از این تابع می‌توان حافظه مشترک را حذف یا مجوزهای دسترسی آن را تغییر داد.

```
int shmctl(int id, int cmd, struct shmid_ds *buf);
```

- **id**: شناسه حافظه مشترک.
- **cmd** عملیات مورد نظر را مشخص می‌کند.
- **IPC_RMID:** این دستور حافظه مشترک و شناسه آن را از سیستم حذف می‌کند.
- **IPC_SET:** برای تغییر مالکیت یا قوانین دسترسی حافظه مشترک.
- **IPC_STAT:** اطلاعات حافظه مشترک را در ساختار داده‌ای buf ذخیره می‌کند.
- این تابع در صورت موفقیت مقدار ۰ برمی‌گرداند.

```c
int main() {
    key_t key = IPC_PRIVATE;
    int shm_id = shmget(key, 1024, IPC_CREAT | S_IRUSR | S_IWUSR);
    if (shm_id == -1) {
        perror("Failed to create shared memory");
        exit(1);
    }

    char *shared_memory = (char *)shmat(shm_id, NULL, 0);
    if (shared_memory == (char *)-1) {
        perror("Failed to attach to shared memory");
        exit(1);
    }

    strcpy(shared_memory, "Hello, Shared Memory!");

    printf("Message from shared memory: %s\n", shared_memory);

    if (shmdt(shared_memory) == -1) {
        perror("Failed to detach from shared memory");
        exit(1);
    }

    if (shmctl(shm_id, IPC_RMID, NULL) == -1) {
        perror("Failed to remove shared memory");
        exit(1);
    }

    return 0;
}
```

ایجاد کلید حافظه مشترک:

IPC_PRIVATE به معنی این است که بخش حافظه مشترک جدیدی فقط برای این برنامه ایجاد می‌شود و سایر فرآیندها نمی‌توانند به آن دسترسی پیدا کنند مگر اینکه شناسه آن را بدانند.

ایجاد حافظه مشترک:

- shmget تابعی است که برای ایجاد یا دسترسی به بخش حافظه مشترک استفاده می‌شود.
- key مشخص می‌کند که آیا حافظه جدیدی ایجاد شود یا به حافظه‌ای موجود متصل شود.
- ۱۰۲۴ اندازه حافظه مورد نیاز را بر حسب بایت مشخص می‌کند.
- IPC_CREAT | S_IRUSR | S_IWUSR مجوزهایی را تنظیم می‌کند. S_IRUSR به کاربر اجازه خواندن و S_IWUSRاجازه نوشتن می‌دهد.
- اگر shmget مقدار -۱ را برگرداند، به این معنی است که حافظه مشترک ایجاد نشده و برنامه یک پیام خطا نشان داده و از برنامه خارج می‌شود.

```c
int main() {
    key_t key = IPC_PRIVATE;
    int shm_id = shmget(key, 1024, IPC_CREAT | S_IRUSR | S_IWUSR);
    if (shm_id == -1) {
        perror("Failed to create shared memory");
        exit(1);
    }

    char *shared_memory = (char *)shmat(shm_id, NULL, 0);
    if (shared_memory == (char *)-1) {
        perror("Failed to attach to shared memory");
        exit(1);
    }

    strcpy(shared_memory, "Hello, Shared Memory!");

    printf("Message from shared memory: %s\n", shared_memory);

    if (shmdt(shared_memory) == -1) {
        perror("Failed to detach from shared memory");
        exit(1);
    }

    if (shmctl(shm_id, IPC_RMID, NULL) == -1) {
        perror("Failed to remove shared memory");
        exit(1);
    }

    return 0;
}
```

اتصال حافظه مشترک به فضای آدرس فرآیند:

- shmat برای اتصال بخش حافظه مشترک به فضای آدرس فرآیند استفاده می‌شود.
- shm_id شناسه حافظه مشترک است که در مرحله قبل به دست آوردیم.
- اگر shmat مقدار -۱ برگرداند، به معنی شکست در اتصال است و برنامه یک پیام خطا چاپ کرده و از برنامه خارج می‌شود.

نوشتن پیام در حافظه مشترک:

- از strcpy برای کپی کردن رشته "Hello, Shared Memory!" به حافظه مشترک استفاده می‌شود.

خواندن و چاپ پیام از حافظه مشترک

```c
int main() {
    key_t key = IPC_PRIVATE;
    int shm_id = shmget(key, 1024, IPC_CREAT | S_IRUSR | S_IWUSR);
    if (shm_id == -1) {
        perror("Failed to create shared memory");
        exit(1);
    }

    char *shared_memory = (char *)shmat(shm_id, NULL, 0);
    if (shared_memory == (char *)-1) {
        perror("Failed to attach to shared memory");
        exit(1);
    }

    strcpy(shared_memory, "Hello, Shared Memory!");

    printf("Message from shared memory: %s\n", shared_memory);

    if (shmdt(shared_memory) == -1) {
        perror("Failed to detach from shared memory");
        exit(1);
    }

    if (shmctl(shm_id, IPC_RMID, NULL) == -1) {
        perror("Failed to remove shared memory");
        exit(1);
    }

    return 0;
}
```

جدا کردن حافظه مشترک از فضای آدرس:

- shmdt برای جدا کردن حافظه مشترک از فضای آدرس فرآیند استفاده می‌شود.
- اگر shmdt مقدار -۱ برگرداند، به معنی شکست در جدا کردن حافظه است و برنامه یک پیام خطا چاپ کرده و از برنامه خارج می‌شود.

حذف حافظه مشترک از سیستم:

- shmctl برای کنترل حافظه مشترک استفاده می‌شود. با استفاده از IPC_RMID این حافظه از سیستم حذف می‌شود.
- اگر shmctl مقدار -۱ برگرداند، به معنی شکست در حذف حافظه مشترک است و برنامه یک پیام خطا چاپ کرده و از برنامه خارج می‌شود.