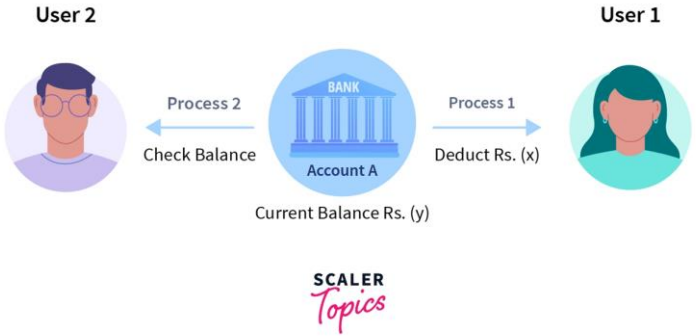
پروژه چهارم آزمایشگاه سیستم عامل

همگام سازی

همگام سازی پردازه ها در سیستم عامل

- اجرای همزمان پردازه ها توسط سیستم عامل اهمیت بسیار زیادی دارد، چرا که کاربر نیاز به این همگام سازی دارد. این همگام سازی زمانی مهمتر می شود که این پردازه ها از منابع مشترک استفاده کنند. پس سیستم عامل باید بتواند این منابع را به خوبی مدیریت کند.



قفل

- قفل یک مکانیزم همگام سازی است تا دسترسی به منابع در سیستم عامل را توسط پردازه ها محدود کند.
- از قفل برای جلوگیری ورود پردازه ها به ناحیه بحرانی استفاده می شود.

انواع قفل در XV6

spinlock.h

```
// Mutual exclusion lock.
struct spinlock {
  uint locked;       // Is the lock held?

  // For debugging:
  char *name;        // Name of lock.
  struct cpu *cpu;   // The cpu holding the lock.
  uint pcs[10];      // The call stack (an array of program counters)
                     // that locked the lock.
};
```

قبل از ناحیه های بحرانی در کد استفاده شده است و هر پردازه زمانی که به spinlock آن نقطه بحرانی برسد، وارد یک حلقه می شود و هر بار چک میکند که آیا قفل آزاد شده است یا خیر.

sleeplock.h

```
// Long-term locks for processes
struct sleeplock {
  uint locked;       // Is the lock held?
  struct spinlock lk; // spinlock protecting this sleep lock

  // For debugging:
  char *name;        // Name of lock.
  int pid;           // Process holding lock
};
```

برخلاف spinlock که پردازه را به صورت تمام مدت مشغول نگه می دارد (busy waiting) ، پردازه را به حالت sleeping میبرد و زمانی که نقطه بحرانی قابل دسترسی باشد، سیستم عامل به صورت رندوم یکی از این پردازه ها را بیدار می کند.

سمافور

- قفلی هایی تا الان دیدیم همگی mutex بودند.
- برخلاف mutex که می تواند تنها به یک پردازه اجازه دسترسی به ناحیه بحرانی بدهد، سمافور با استفاده از متغیر شمارشی خود می تواند تا سقف تعداد پردازه ای که برای آن مشخص می شود، اجازه دسترسی به پردازه ها را می دهد.
- برای استفاده از semaphore باید از رابط های زیر استفاده کنید:
  - *sem_init(i,v)*
  - *sem_acquire(i)*
  - *sem_release(i)*

مسئله readers-writers

فرض کنید یک پایگاه داده وجود دارد که تعدادی پردازه به صورت همزمان از آن استفاده می کنند. برخی می خواهند آن را بخوانند و برخی آن را بروزرسانی کنند:

- طبیعتا هنگامی که چند پردازه همزمان فقط دسترسی خواندن دارند، مشکلی پیش نمی آید.
- زمانی مشکل به وجود می آید که یک پردازه با دسترسی نوشتن به داده ها وجود داشته باشد. پس این پردازه ها باید دسترسی مناسبی به داده های اشتراکی داشته باشند.

این مسئله انواع مختلفی دارد که همگی اولویت هایی دارند. یکی از این مدل ها، مدل اولویت به reader ها است که تا زمانی که یک پردازه درخواست خواندن دارد، به هیچ writer ای اجازه دسترسی به ناحیه بحرانی داده نشود. writer ها زمانی اجازه دسترسی پیدا می کنند که هیچ reader ای درخواست دسترسی نداشته باشد.
برای آن که چندین reader همزمان بتوانند از داده های اشتراکی استفاده کنند، باید از سمافور استفاده کرد.

موفق باشید

امیرمهدی جودی
علی عباسی