## مسئله وارونگی اولویت (Priority Inversion)

سناریویی در زمانبندی است که یک فرآیند با اولویت پایین به طور غیرمستقیم موجب مسدود شدن یک فرآیند با اولویت بالاتری شود.

برای مثال به سناریویی که در ادامه آورده شده است دقت نمایید. فرض کنید در سیستمی دو فرآیند $P_H$ (فرآیند با اولویت بالا) و $P_L$ (فرآیند با اولویت پایین) موجود است به گونه‌ای که هر دو فرآیند مذکور در تلاش برای تصاحب انحصاری منبع R هستند. اگر $P_L$ منبع R را در اختیار داشته باشد و فرآیند $P_H$ تلاش برای بدست آوردن منبع R کند به سبب درگیری انحصار متقابل فرآیند $P_H$ مسدود می‌شود (در حالتیکه از کنفیگ مسدود به جای انتظار مشغول استفاده می‌کنیم). لازم به ذکر است تا زمانیکه فرآیند $P_L$ منبع R را آزاد نکند، فرآیند $P_H$ در وضعیت مسدود باقی خواهد ماند.

حال فرض کنید که یک فرآیند سوم به نام $P_M$ که دارای اولویت متوسط نسبت به دو فرآیند مذکور است در چنین شرایطی به مجموعه فرآیندها اضافه شود. لازم به ذکر است که

فرض ما در اینجا آن است که فرآیند $P_M$ وابسته به منبع R نیست. در این صورت ارجحیت اجرای فرآیند $P_M$ از $P_L$ بالاتر بوده و لذا سبب می شود اجرای $P_L$ متوقف شده و اجرای فرآیند $P_M$ با الویت آن پایین تر از $P_H$ است در دستور کار گیرد.
قرار

در واقع فرآیند $P_H$ دچار یک اشتداد غیرقابل پیش بینی می شود که به طور غیرمستقیم توسط یک کار با ارجحیت پایین تر نظیر M ایجاد شده است.

مثال عملی: مشکلی که برای کاوشگر مریخ به نام Mars PathFinder در سال 1997 حادث شد به سبب وارونگی ارجحیت (الویت) در سیستم های بلادرنگ تشخیص داده شده است.

تمرین: آیا راه حل ارائه شده توسط سمافور عمومی برای مسأله ناحیه بحرانی، مشکل وارونگی الویت را دارد یا خیر؟ تحلیل کنید.

## (۴۴) مثال هایی از همگام سازی با استفاده از سمافورها:

مثال ۱: دو فرآیند $P_1$ و $P_2$ را در نظر بگیرید که می خواهید ممرو متری در آنها به نحوی باشد که ابتدا دستور $S_1$ در فرآیند $P_1$ و سپس دستور $S_2$ در فرآیند $P_2$ اجرا شود.

Semaphore S; S.count = 0

| $P_1$ | $P_2$ |
|---|---|
| ⋮ | ⋮ |
| $S_1$; | wait(S); |
| Signal(S) | $S_2$; |
| ⋮ | ⋮ |

مشخص است که تا زمانیکه دستور $S_1$ از فرآیند $P_1$ اجرا نشود، اجرای $S_2$ ممکن نیست.

مثال ۲: آیا ترتیب اجرای $P_1, P_2, P_3$ (از چپ به راست) با مقدار اولیه‌های $S.count=0$ و $Q.count=0$ امکان‌پذیر است؟

| $P_1$ | $P_2$ | $P_3$ |
|---|---|---|
| ⋮ | wait(Q); | wait(S); |
| ⋮ | ⋮ | ⋮ |
| signal(S); | | signal(Q); |

فرآیند $P_1$ بدون هیچ مانعی اجرا شده و سرانجام سافورد S را نیز یکی می‌کند ($S.count=1$)، از آنجایی که $Q.count=0$ است فرآیند $P_2$ نمی‌تواند اجرای خود را کامل کند و تنها هنگامی که فرآیند $P_3$ به طور کامل اجرا شود امکان اجرای فرآیند $P_2$ فراهم خواهد شد چرا که سرانجام با پایان اجرای فرآیند $P_3$ مقدار $Q.count=1$ می‌شود.

در نتیجه امکان اجرای $P_1P_2P_3$ وجود ندارد و پاسخ منفی است.

مثال ۳: با فرض اینکه مقدار اولیه دو سافورد S و Q برابر صفر است، نحوه چاپ ABCDE را مشخص کنید.

```
Semaphor S,Q;
S.count=Q.count=0;
```

| $P_0$ | $P_1$ |
|---|---|
| signal(Q); | wait(Q); |
| wait(S); | cout<<"A"; |
| cout<<"C"; | signal(S); |
| cout<<"D"; | cout<<"B"; |
| | cout<<"E"; |

پاسخ: در زیر ترتیب اجرا با شماره گذاری دستورات فرآیندها نشان داده شده است.

```
Semaphore S,Q;
S.Count = Q.Count = 0;
```

| #line | $P_0$ | #line | $P_1$ |
|---|---|---|---|
| 1 | signal(Q); | 2 | wait(Q); |
| 6 | wait(S); | 3 | cout << "A"; |
| 7 | cout << "C"; | 4 | signal(S); |
| 8 | cout << "D"; | 5 | cout << "B"; |
| | | 9 | cout << "E"; |

مثال ۲: نحوه اجرای فرآیندهای $P_0$ و $P_1$ چگونه باشد تا مقدار نهایی برابر 6 شود. a=10, b=6

```
Semaphor S; S.Count = 1;
int a,b; a=b=1;
```

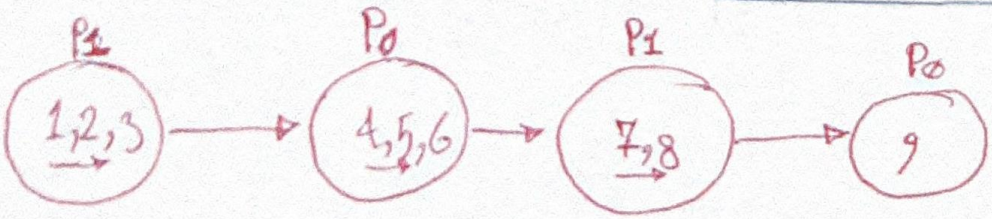
| line# | $P_0$ | line# | $P_1$ |
|---|---|---|---|
| 4 | a = a+2; | 1 | a = 3; |
| 5 | b = b+1; | 2 | wait(S); |
| 6 | signal(S); | 3 | b = a+b; |
| 9 | b = b+1; | 7 | wait(S); |
| | | 8 | a = a+b; |

S=1, a=5, b=5

S=0, a=10, b=6

S=0, a=3, b=4

S=0, a=10, b=5



مثال ۵: با فرض اینکه مقدار اولیه سمافور S برابر ۸ (S.count = 8)، سمافور P برابر ۳ (P.count = 3) و سمافور Q برابر ۱ (Q.count = 1) باشد حداکثر چند فرآیند مشخص هر سمافور را آزاد می‌کند.

```
    .
    .
    .
Wait(S);
Wait(P);
Wait(Q);
    .
    .
    .
signal(Q);
signal(P);
signal(S);
    .
    .
    .
```

پاسخ: فرض کنید n فرآیند در سیستم وجود دارد، طبق توضیحات بالا، حداکثر ۸ فرآیندی که از Wait(S) عبور کنند و بقیه یعنی n-8 فرآیند در صف سمافور S می‌خوابند. از ۸ فرآیندی که از سمافور S عبور کرده‌اند، حداکثر ۳ فرآیند می‌توانند از سمافور P عبور کنند و ۵ فرآیند در صف P می‌خوابند. در نهایت نیز از ۳ فرآیندی که از سمافور P عبور کرده‌اند، حداکثر ۱ فرآیند از Wait(Q) عبور کرده و ۲ فرآیند در صف Q می‌خوابند.