تمرین سوم

|  |  |
| --- | --- |
| علی علی‌محمّدی | 9613027 |

|  |  |
| --- | --- |
| 1) | از معایب غیرفعال‌کردن وقفه‌ها می‌توان به موارد زیر اشاره کرد:   * غیرفعال‌کردن وقفه‌ها اخطارهای وقایع خارجی را که می‌توانند باعث انجام Context Switching شوند (مانند Timer)، مسدود می‎کند. * در یک سیستم واقعی، غیرفعال‌کردن وقفه‌ها تنها در اختیار هسته (Kernel) قرار دارد و یک دستور privileged است و کاربر به آن دسترسی ندارد. * غیرفعال‌کردن وقفه‌ها در یک سیستم چندپردازنده‌ای (یا دارای چند هسته‌ی پردازنده) ناکارآمد است.   Test\_and\_Set یک دستورالعمل ویژه‌ی زبان اَسمبلی است که دو عملیات را به صورت مستقل انجام می‌دهد (به این معنی که دستورالعمل نمی‌تواند در میانه‌ی اجرا دچار وقفه شود) و در عین حال لازم نیست وقفه‌ها غیرفعال شوند.  در سیستم‌هایی که تنها یک پردازنده (یا یک هسته‌ی پردازشی) وجود دارد، برای حل مسئله‌ی ناحیه‌ی بحرانی از روش غیرفعال‌کردن وقفه‌ها استفاده می‌شود. |
| 2) | زمان اجرای برنامه 10 ثانیه خواهد بود.  #include <stdio.h>  #include <pthread.h>  #include <unistd.h>  #include <semaphore.h>  #define M 30  #define N 3  sem\_t mutex;  void\* f(void \*args) {    // Wait  sem\_wait(&mutex);    // Critical  sleep(1);  printf("\*\n");    // Signal  sem\_post(&mutex);  }  int main (int argc, char \*argv[]) {    sem\_init(&mutex, 0, N);    pthread\_t threads[M];    for(int i=0; i<M; i++)  pthread\_create(&threads[i], NULL, f, NULL);    for(int i=0; i<M; i++)  pthread\_join(threads[i], NULL);    sem\_destroy(&mutex);    return 0;  } |
| 3) | خروجی برنامه 2000000 خواهد بود.  #include <stdio.h>  #include <pthread.h>  #include <semaphore.h>  long x = 0;  sem\_t mutex;  void \*thread() {  // Wait  sem\_wait(&mutex);    // Critical  for(int i=0; i<1000000; ++i)  x++;    // Signal  sem\_post(&mutex);  }  int main() {  sem\_init(&mutex, 0, 1);  pthread\_t tid\_1, tid\_2;  pthread\_create(&tid\_1, NULL, thread, NULL);  pthread\_create(&tid\_2, NULL, thread, NULL);  pthread\_join(tid\_1, NULL);  pthread\_join(tid\_2, NULL);  sem\_destroy(&mutex);  printf("%ld\n", x);  return 0;  } |
| 4) | قحطی‌زدگی یا مسدود کردن نامحدود پدیده‌ای است که به الگوریتم‌های زمان‌بندی اولویت مربوط است و در آن، یک فرایند که منتظر است تا در CPU اجرا شود، به دلیل اولویت پایین ممکن است به طور نامحدود صبر کند. در سیستم رایانه‌ای که بار پردازشی زیادی دارد، جریان مداوم فرایندهای با اولویت بالاتر می‌تواند مانع از آن شود که یک فرایند با اولویت کم بتواند به CPU دسترسی پیدا کند.  قحطی‌زدگی ممکن است به دلیل اشتباه در الگوریتم زمان‌بندی یا اشتباه در الگوریتم حذف متقابل (Mutual Exclusion) باشد. همچنین می‌تواند توسط نشت منابع (Resource Leaks) ایجاد شود یا که می‌تواند عمداً از طریق یک حمله‌ی محروم‌سازی از سرویس مانند بمبِ چنگال (Fork Bomb) ایجاد شود.  الگوریتم‌های اول کوتاه‌ترین کار (Shortest Job First) و زمان‌بندی مبتنی بر اولویت (Priority-based Scheduling) می‌توانند باعث قحطی‌زدگی شوند. مثال دیگری از الگوریتم‌هایی که می‌تواند باعث قحطی‌زدگی شود، می‌توان الگوریتم زمان‌بندی حداکثر توان عملیاتی (Maximum Throughput Scheduling) در شبکه‌های بی‌سیم را نام برد.  سال‌خوردگی روشی برای افزایش تدریجی اولویت فرایندهایی است که مدت زیادی در سیستم منتظر می‌مانند. به عنوان مثال، اگر اولویت از (کم) تا (زیاد) باشد، می‌توانیم اولویت فرایند در حال انتظار را هر ثانیه، یک واحد افزایش دهیم. به این ترتیب، حتی فرایندی با اولویت اولیه‌ی پس از گذشت حدود دو دقیقه به فرایند با اولویتِ تبدیل می‌شود. |
| 5) | **FCFS**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | P5 | P4 | P3 | P2 | P1 |   **RR**   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | P2 | P5 | P2 | P5 | P4 | P3 | P2 | P1 |   **SJF**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | P2 | P5 | P1 | P4 | P3 |   به این ترتیب مشاهده می‌شود که الگوریتم SJF دارای کمترین میانگین زمان انتظار است. |
| 6) | نمودار اجرای فرایندها به شکل زیر است:   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | P1 | P4 | P1 | P4 | P1 | P4 | P3 | P1 | P3 | P1 | P3 | P2 | P1 | P2 | P1 | P2 | P1 |   به این ترتیب زمان اتمام همه‌ی فرایندها به دست می‌آید:   |  |  |  | | --- | --- | --- | | زمان اتمام (میلی‌ثانیه) | زمان ورود (میلی‌ثانیه) | فرایند | | 19 | 0 | P1 | | 7 | 1 | P2 | | 12 | 5 | P3 | | 17 | 10 | P4 |   زمان برگشت و زمان انتظار هر فرایند با استفاده از روابط زیر قابل محاسبه است:   |  |  | | --- | --- | |  |  |   با استفاده از روابط فوق، زمان برگشت و زمان انتظار همه‌ی فرایندها به دست می‌آید:   |  |  |  | | --- | --- | --- | | زمان انتظار (میلی‌ثانیه) | زمان برگشت (میلی‌ثانیه) | فرایند | | 9 | 19 | P1 | | 3 | 6 | P2 | | 4 | 7 | P3 | | 4 | 7 | P4 |   اکنون با استفاده از جدول فوق میانگین زمان برگشت و میانگین زمان انتظار قابل محاسبه است:  به این ترتیب، میانگین زمان برگشت 9.75 میلی‌ثانیه و میانگین زمان انتظار 5 میلی‌ثانیه است. |