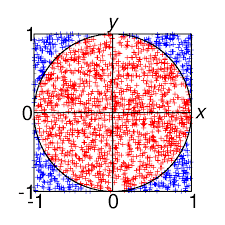
|  |  |  |
| --- | --- | --- |
|  | PAMUKKALE ÜNİVERSİTESİ**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ** **CENG 305 – İşletim Sistemleri 2018-2019 EĞİTİM ÖĞRETİM YILI GÜZ DÖNEMİ ARASINAVI** |  |
| **Adı Soyadı:** | **Numarası:** | **İmza:** |

**15.11.2018 Süre:90 dk**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Soru** | **1 (10p)** | **2 (15p)** | **3 (15p)** | **4 (25p)** | **5(20p)** | **6(15p)** |
| **Puan** |  |  |  |  |  |  |

1. Eski işletim sistemleri sonradan değiştirilemeyen bellek alanlarında (ROM) tutulmaktaydı. Bu yöntemin avantaj ve dezavantajlarını tartışınız.
2. Bir process içerisindeki threadler hangi alanları paylaşıp, hangi alanlar kendilerine özel kullanılmaktadır? Şekil üzerinde açıklayabilirsiniz.

1. Processler arası iletişim ne şekillerde gerçekleştirilebilir? Örneklerle gösteriniz.
2. (**Multi-threaded**) Bu soruda Java threadlerini kullanarak bir multi-threaded program yazmanız istenmektedir. Pi sayısını yaklaşık olarak hesaplamak için birim çember kullanılabilir. Birim çember yarıçapı 1 olan dairedir. Bu yöntemde x ve y koordinatlarını [-1,1] aralığında rasgele seçip bu noktaların birim çember içine düşenleri (dairinenin içine), toplamda üretilen nokta sayısıyla oranladığınızda π/4 sayısı yaklaşık olarak elde edilebilir. Bu şekilde ne kadar çok rastgele nokta üretirseniz π sayısına yakınsamanız artacaktır. Bu yöntemi kullanarak π sayısını hesaplayan mantıklı bir multi-threaded bir Java uygulaması yazınız.



1. Tek bir CPU bulunan sistemde aşağıdaki processleri çalıştırdığımızı düşünelim.

|  |  |  |  |
| --- | --- | --- | --- |
| Process | Kuyruğa Gelme Zamanı | Çalışma Süresi | Öncelik (Priority) |
| P1 | 0 | 10 | 2 |
| P2 | 2 | 8 | 1 |
| P3 | 3 | 3 | 3 |
| P4 | 10 | 4 | 2 |
| P5 | 12 | 1 | 3 |
| P6 | 15 | 4 | 1 |

1. FCFS scheduling için Gantt chart, ortalama turnaround (çevrim) süresini ve ortalama waiting süresini gösteriniz.

1. Non-preemptive SJF (Shortest Job first) için Gantt chart, ortalama turnaround (çevrim) süresini ve ortalama waiting süresini gösteriniz.
2. Preemptive SJF için Gantt chart çizip, ortalama turnaround (çevrim) süresini ve ortalama waiting süresini gösteriniz.
3. Preeemptive Priority scheduling algoritması için Gantt chart, ortalama turnaround (çevrim) süresini ve ortalama waiting süresini gösteriniz.
4. Aşağıdaki kodun çıktısı ne olur? Açıklayınız.

#include <stdio.h>

#include <stdlib.h>

**int** main ()

{

**int** pid,pid2,pid3;

printf ("I'm the original process with PID %d and   
 PPID %d.\n",getpid (), getppid ());

pid = fork ();

pid2 =fork();

execl("/bin/ls", "ls", "-l", **NULL**);

pid3=fork();

printf ("PID %d terminates.\n", getpid () );

**return** 0;

}

**Başarılar**