INF114 Algoritma ve Ileri Programlama

Quiz03 Dinamik Bellek ve Queue-Stack 08/05/2023

- Tüm fonskiyonları tanımladıktan sonra `main.c` dosyasında hazırlanmış kod ile deneyiniz, düzgün çalıştığından emin olunuz. Yazdığınız program hatasız derlendiği sürece gidişata göre puan alacaksınız. Derleme hatası veren programlara, yazılan koda bakılmaksızın sıfır verilecektir.
- Kodun okunabilirliği (girintileme, değişken isimleri, vs...) 10 puan üzerinden değerlendirilecektir.
- `process.h, process.c, main.c` dosyalarını zipleyip `Quiz03_Isim_Soyisim.zip` formatında Moodle'a yükleyiniz. Örneğin, `Quiz03_Onur_Ozcelik.zip`.
- **En Geç Teslim Tarihi: 12/05/2023 Cuma 23:55**

Sorular

```
1  struct process {
2      int PID; // process id
3      int dur; // process duration
4      int is_interrupt; // whether the process is an ↩
           interruption or not.
5  };
6
7  struct process_queue {
8      struct process process;
9      struct process_queue *next;
10 };
```

Quiz'de dinamik olarak queue (kuyruk) ve stack (yığıt) veri yapılarını birlikte kodlayacağız ve basite indirgenmiş bir CPU (Central Process Unit)'nun process kuyruğunu round_robin çizelgeleyici ile simüle edeceğiz. Kuyruk veri yapısı, LILO (Last-In-Last-Out) veya başka bir deyişle FIFO (First-In-First-Out) ilk giren ilk çıkar prensibine sahip veri yapısıdır.

Tanımlayacağımız process (süreç) kuyruğu iki tür process barındıracak. Bunu, process yapısının içerisindeki is_interrupt değeri gösterecek. Eğer bu değer 1 ise process kesmedir (interrupt); değilse standart bir processstir. Bu ikisi arasındaki ayrım herhangi bir process kesme (interrupt, i.e., is_interrupt=1) ise bu process kuyruğun en başında aksi takdirde kuyruğun sonuna sıraya eklenecek.

Moodle üzerinde verilen "Quiz03.zip" arşiv dosyası içindekileri kendi projenize ekleyiniz. Arşiv içinde 3 tane programlama dosyası bulunuyor. Bu dosyalar `process.h, process.c, main.c`.

- `process.c` içerisindeki gerekli fonksiyonları tanımlayınız.
  - `int is_empty(struct node* head)` Verilen kuyruğun boş olup olmadığını döndürür. Kuyruk boşsa 0, değilse 1 döndürmelidir.
  - `void enqueue(struct process_queue**, struct process)` Verilen process bilgilerini taşıyan yeni process oluşturulup kuyruğun sonuna eklenir.
  - `void push(struct process_queue** top, struct process)` Verilen process bilgilerini taşıyan yeni bir process oluşturup kuyruğun başına eklenmelidir. Yani kuyruğun en üstündeki değer (top) yeni eklenen eleman olacak şekilde güncellenmelidir.
  - `void dequeue_and_pop(struct process_queue** head)` Verilen kuyruğun en başındaki (head) eleman kuyruktan silinir.
  - `struct process_queue* peek(struct process_queue* head)` Verilen kuyruğun başındaki process'i yani işlenecek bir sonraki process'in pointerı döndürülür (burada free yapılmacak).
  - `void round_queue(struct process_queue** head)` Kuyruğun başındaki eleman (head) kuyruğun sonu olacak şekilde döndürülür. Bu takdirde kuyruğun başındaki eleman ötelenecek ve sonundaki eleman ise önceki head olacak.
    - Eğer kuyrukta tek bir process varsa herhangi bir işleme gerek olmayacak.
    - Kuyruğun sonuna aktarılacak en baştaki (head) process kuyruğun sonuna aktarıldığında next'i NULL gösterecek şekilde güncellenmelidir.
- `main.c` içerisindeki `void round_robin(struct process_queue* head, int pid)` fonksiyonunu tanımlayınız. Burada sadece T0DO ile belirtilen kısımları yapmanız gerekiyor.

  Round-Robin çizelgeleyici standart (interrupt/kesme olmayan) process'lerin belli sürelerle işletilip tamamlandığı takdirden kuyruktan çıkarıldığı tamamlanmadığı takdirde kuyruğun sonuna aktarıldığı bir çizelgeleme algoritmasıdır. Kesme process olduğu zaman başka bir kesme olmadığı takdirde process tamamlanıla kadar işletilir. Kesme gelirse en son gelen kesme işletilir. Özetle, standart process'ler kuyruk (FIFO); kesmeler ise stack (LIFO) mantığıyla işletilir ve Round-Robin algoritması uyarınca standart process'ler parça parça işletilir.

  Tasarlanan yapıda standart process'ler (10, 20, 30)ms; kesmeler ise (10, 20)ms süreleri arasından birine sahip olurlar ve her bir process için her döngüde geçen süre 10ms olarak düşünülüyor. Yani 30ms'lik standart bir process kesme gelmediği sürece 3 döngüde tamamlanır. round_robin fonksiyonu da içerisindeki kuyruktaki processlerin her biri tamamlanana kadar dönmektedir.