INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA TRIÂNGULO MINEIRO	Curso: Bacharelado em Ciência da Computação		
	Unidade Curricular: Sistemas Operacionais	Ano/Período:	2022/ 7
	Tipo de Atividade: Pratica #2	Data:	04/05/2022
	Professor: Getúlio de Morais Pereira	Valor:	Entrega: 04/05/2022

Contextualização:

No sistema operacional, cada processo/thread ativo possui um código de identificação único, definido como PID (*proccess identifier*). Geralmente, esse PID varia num intervalo de valores inteiros, por exemplo na faixa entre 300 e 5000 (incluindo extremos).

O gerenciador de PIDs do sistema operacional é o módulo responsável por gerenciar os valores usados como identificadores de processos/threads. Quando um processo/thread é criado, o gerenciador de PIDs atribui a ele um valor de PID **exclusivo**. Quando o processo/thread termina, o valor do PID é retornado ao gerenciador de PIDs, que pode atribuí-lo novamente mais tarde, para outro processo/thread. Os identificadores de processo são discutidos com mais detalhes na Seção 3.3.1. de [1]. O mais importante nesse ponto é reconhecer que os identificadores de processo/thread devem ser exclusivos, ou seja, dois processos/threads ativos não podem ter o mesmo PID.

Uma forma de definir, em linguagem de programação, um intervalo de valores inteiros para uso como PIDs é por meio de constantes, como na listagem 1

Listing 1: Definição de faixa de valores para PIDs (em C-ANSI)

```
#define MIN_PID 300
#define MAX_PID 5000
```

O gerenciador de PIDs utiliza uma estrutura de dados para armazenar esses valores de uso exclusivo. Uma estratégia adotada pelo Linux é implementar esse repositório como um mapa de bits, onde o valor 0 na posição i do mapa indica que um ID de processo/thread de valor i está disponível, e um valor 1 indica que o ID de processo/thread está em uso no momento.

1. Implemente a API listada abaixo, cujas funcionalidades são descritas nos respectivos comentários.

Listing 2: A estrutura do processo produtor (reprodução da fig 5.9)

```
/*
    Cria e inicializa uma estrutura de dados para representar PIDs;
    Retorna: -1, se não for bem-sucedida, e
             1, se for bem-sucedida.
*/
int allocate_map (void);

/*
    Aloca e retorna um PID;
    Retorna: -1 se for incapaz de alocar um pid (todos os PIDs estão em uso),
             caso contrário, o valor do ID alocado
*/
int allocate_pid (void);
```

```
/*
    Libera um PID
*/
void release_pid (int pid);
```

2. Implemente um programa que teste a API solicitada na questão 1. Esse programa criar múltiplas threads (por exemplo, 100) onde cada thread solicitará um PID, entrará em suspensão por um período de tempo aleatório e, então, liberará o PID. A entrada em suspensão por período de tempo aleatório é semelhante ao uso típico do PID em threads reais nos sistemas operacionais: um PID é atribuído a um novo processo/thread; o processo/thread é executado e encerrado; e o PID é, então, liberado no encerramento do processo/thread). Em sistemas UNIX e Linux, a suspensão é obtida pela função *sleep( )*[1], que recebe um valor inteiro representando quantos segundos durará a suspensão.

Referências

[1] Abraham Silberschatz; Peter Baer Galvin; Greg Gagne. Fundamentos de Sistemas operacionais. LTC, 9 edition, 2015.

[1] existem um método correspondente em Java