Relatório T3 SO - Andriza Campanhol

- ## Implementação das filas

As filas foram criadas através da estrutura de lista encadeada, onde temos uma lista para cada um dos estados do simulador. As listas estão inicialmente vazias e são posteriormente preenchidas para apontar para o primeiro processo de cada lista, que consequentemente deverá apontar para o próximo processo.

```
struct list{
    PCB* inicio;
};
typedef struct list List;
```

As filas criadas são: ***listNew***, ***listReady***, ***listRunning***, ***listExit*** e ***listProc***.

- ## Estrutura de dados do processo PCB

O PCB foi implementado através de uma *struct* que define os dados que serão guardados sobre cada um dos processos.

```
struct pcb{
    int id;
    char* status;
    int tamVet;
    int* vet;
    struct pcb* prox;
    struct pcb* proxProc;
};
typedef struct pcb PCB;
```

- ➔ O ***id*** corresponde ao identificador(PID) no PCB, seu valor é dado pela ordem de criação dos processos (contador de processos);
- ➔ O ***status*** corresponde ao estado em que o processo se encontra (“new”, “ready”, “running”...);
- ➔ O ***tamVet*** corresponde ao tamanho do vetor que guarda os tempos de CPU e dispositivos do processo;
- ➔ O ***vet*** corresponde ao vetor de CPU e dispositivos do processo;
- ➔ O ponteiro ***prox*** aponta para o próximo processo da fila a qual ele pertence no momento;
- ➔ O ponteiro ***proxProc*** corresponde ao próximo processo em ordem de criação dos processos (ordenado).

- ## Estrutura do simulador

Primeiramente, são abertos os arquivos para a leitura da entrada e para, posteriormente, a escrita da saída. Então, são criadas as filas (inicialmente vazias) e acontece a leitura da primeira linha de entrada, onde

são guardados os valores da quantidade de processos e dispositivos, assim como um vetor contendo os tempos dos dispositivos.

Em sequência, é lido o tempo do primeiro processo e começa a contagem dos tempos do simulador, controlado por um *while* com a condição de parada de que quando a quantidade de processos for igual a quantidade de processos na fila *exit*, então a simulação chegou ao fim.

Quando o tempo for igual ao tempo em que o processo chega, é criado um PCB para o processo e ele é adicionado na fila *new* e na fila de processos ordenados (*listProc*), depois, lê-se o tempo de chegada do próximo processo e aumenta-se o contador de processos. Caso o próximo processo chegue ao mesmo tempo já será lido, senão, será criado quando o simulador atingir seu tempo.

Enquanto a fila *ready* tiver quantidade menor ou igual a 2 e a fila new não estiver vazia, os processos passarão de *new* para *ready*.

Se a fila *running* estiver vazia e existir algum processo na fila *ready*, o primeiro processo pronto passará para *running*, onde ficará pela quantidade de CPU que precisar na primeira posição do vetor ou uma fatia de 4 tempos. Caso o processo tenha terminado a quantidade de ciclos que precisa nessa fatia, será movido para a fila *exit*, caso contrário, será movido para o final da fila *ready* novamente.

Nas mudanças de fila, são atualizados os ponteiros das filas e o estado de cada processo. No final de cada tempo, é escrito no arquivo o tempo e o estado dos processos ordenadamente. Ademais, são utilizadas variáveis controladoras para realizar a mudança de um processo de um estado para o outro.