Implementação

O projeto é constituído por 6 ficheiros. O *makefile*, que permite a compilação de ficheiros existentes, um *header "externas.h"* que contem toda a declaração de estruturas necessárias e importação das librarias, para o funcionamento do trabalho, o ficheiro *"servidor.c"*, que inclui o código preciso para execução do projeto. O ficheiro de texto "*config.txt"* contem a informação definida para o arranque do sistema, tais como o numero de processos a criar, numero de threads, tempo de funcionamento dos processos e o tamanho da message queue. Administador.c, aplicação que comunica com o pipe onde entra os pedidos para o sistema. E por fim o ficheiro *"registosSistema.txt"* que será o ficheiro onde fica registado o histórico de entrada e saída do sistema.

O objetivo do trabalho e implementar um simulador das urgências de um hospital, recebendo por um pipe os registos de entrada a tratar, e o sistema gerir os mesmos consoante a prioridade. Quando recebe um sinal SIGUSR1 o sistema imprime as estatísticas processadas do funcionamento do simulador.

Estruturas de Dados

No projeto foram desenvolvidas as seguintes estruturas de dados:

```
typedef struct no* configs;
typedef struct no {
        int threadsTriagem;             // Nº de threads de triagem
        int processosDoutor;                    // Nº de processos de Doutor
        int tempoTurno;                                 //tempo de turno
        int tamanhoFilaAtendimento;     // tamanho da fila de atendimentos
}noConfiguracao;
```

Estrutura usada para guardar os dados que são lidos do ficheiro de configurações, dando inicio a configuração do sistema no arranque. Somente o numero de *threads* pode ser alterado durante a execução do simulador.

```
typedef struct noEstatisticas* estaticas;
typedef struct noEstatisticas{
        long tipoMensagem;
        int numeroPacientesTriados;             //numero total de pacientes triados
        int numeroPacientesAtendidos;   //numero total de pacientes atendidos
        double tempoMedioAntesTriagem;          //tempo medio de espera antes do inicio da
triagem
        double tempoMedioAteAtendimento;        //tempo medio de espera entre o fim da
triagem e o inicio do atendimento
        double mediaTempoTotalUtilizador;       //media do tempo de cada utilizador gastou
desde que chegou ao sistema ate sair
}memoriaPartilhadaSistema;
```

Estrutura usada para a escrita da memoria partilhada do sistema

```
struct queueNO
{
        char nome[20];
        int tempoTriagem;
        int tempoAtendimento;
        int prioridade;
        time_t chegadaSistema;
        time_t tempoInicioQ;
        time_t tempoFimQ;
        time_t inicioAtendimento;
        time_t fimAtendimento;
        double tempoTotal;
        struct queueNO *proximo;
};
```

Estrutura para armazenar a informação da informação da leitura do pipe.

```
struct queue
{
	struct queueNO *primeiro, *ultimo;
};
```

Queue para armazenar cada paciente que de entrada no sistema

```
struct filaMensagens{
	long mtype;
	int tamanhoFila;
	struct queueNO paciente; //tirar ponteiro
}mensagens;
```

Estrutura para armazenar a informação a passar para a *message queue.*

Processo Principal

No arranque do simulador, o processo principal, lê o ficheiro de configurações, cria a memoria partilhada, crias as *threads* e os processos filhos, um de estatísticas e os que vão tratar os pedidos de entrada no sistema.

lePipe

A comunicação com o simulador é efetuada através de um *pipe*, no qual uma *thread* esta sempre há "escuta" dos comandos enviados para o sistema, comandos esse podem ser: ATENDIMENTO=x x x x para entrada de grupos de pacientes ou um simples paciente; TRIAGE=x, alteração de numero de *threads*; estatísticas, impressão no ecrã das estatísticas do sistema; sair, fecha a ligação do processo que comunica com o simulador, deixando o simulador a continuar a fazer o seu trabalho.

servico

As *threads* ao sendo criadas, conforme o parâmetro lido do ficheiro de configurações de arranque do sistema, ficam há espera que na *queue* apareçam pedidos para dar inicio ao seu trabalho. A sua ordem processarem pedidos, e indiferente, dado que a primeira que chegue bloqueia as restantes para que não processem varias o mesmo pedido e não escrevem em memoria partilhada em simultâneo.

atenderPaciente

Quando os processos são criados, são encaminhados para este método. Retiram da *message queue*, um paciente, tratando-o consoante a sua prioridade e o seu tempo de atendimento, escrevendo no final na memoria partilhada a informação final do atendimento do mesmo, descansando, no final ate chegar outro paciente. A gestão dos processos e feita a recurso de sincronização para que dois processos não acedam em simultâneo aos registos da memoria partilhada.

gestorEstatisticas

O gestor de estatísticas, depois de iniciado, fica há espera de um sinal do processo principal para executar, a impressão dos registos do sistema de entrada, medias de tempos efetuados na triagem, no atendimento dos pacientes, que se encontram na memoria partilhada. Cada vez que é ativado um sinal de SIGSR1, imprime no ecrã do sistema as estatísticas do funcionamento do simulador.

Adminstrador

Aplicação usada para testes do simulador. Na aplicação e possível alterar o numero de threads, pedir a informação de estatísticas e inserir no sistema grupos de pacientes ou pacientes simples a serem tratados.

Analises Final

Na execução do trabalho pratico foram usadas em media 280 horas na elaboração do mesmo. Uma estimativa de 5 horas por dia durante os 2 meses de trabalho.