Implementações incorporadas na primeira meta

Autores: Renato Santos 2020134927
Sérgio Alves 2020134949

Código *balcao.c*

O código presente no documento *balcao*.c é responsável pela comunicação entre o programa *balcao* e o programa *classificador*. Também é responsável por obter os valores das variáveis de ambiente "MAXCLIENTES" e "MAXMEDICOS".

O programa começa por cumprir a segunda responsabilidade, dando recurso à função getenv() para adquirir os valores das variáveis anteriormente indicadas e atribuindo-os às variáveis *n* e *m*.

```
char* n = getenv("MAXCLIENTES");
char* m = getenv("MAXMEDICOS");
```

Como ambos os processos precisam de escrever e ler para comunicar, foram abertos dois pipes, de forma a poder haver ambos os tipo de comunicação em ambos os lados.

```
int cp[2]; // Child > Parent
int pc[2]; // Parent > Child

if (pipe(cp) == -1)
{
        printf("Não foi possível criar o pipe \n");
        exit(1);
}
if (pipe(pc) == -1)
```

O fork() cria dois processos, um que vai continuar como *balcao (processo pai)* e outro que vai passar a executar o programa *classificador (processo filho)*.

Como o classificador recebe o seu input pelo stdin e o seu output pelo stdout, tivemos de redirecionar o output e o input dos dois pipes diferentes para o stdin e stdout, de forma a ser estabelecida comunicação entre os processos.

```
if (id == 0) //processo filho
{
        close(STDIN_FILENO);
        dup(pc[0]);

        close(STDOUT_FILENO);
        dup(cp[1]);
```

Enquanto isso o processo pai escreve e lê informação nos polos opostos dos pipes, dando recurso às funções write() e read(), transmitindo e recebendo assim a informação necessária.

```
else //processo pai
{
        close(pc[0]);
        write(pc[1], str, strlen(str) + 1);
        close(pc[1]);

        read(cp[0], ans, 50);
```

Viemos a deparar-nos com um erro que acontecia raramente, em que o processo pai recebia duas linhas do processo filho em vez de apenas uma linha. Para resolver isto, fizemos um for-loop que corta a string depois de uma mudança de linha acontecer. Esta implementação resolveu o erro completamente.

```
for(i=0;ans[i]!='\n';i++)
{
        ans[i] = '\0';

}
```

Header file cliente.h

Esta header file contém uma estrutura de dados simples que guarda apenas duas variáveis. Estas foram as informações que definimos necessárias para o funcionamento do programa *cliente* (sintomas e nome).

```
typedef struct
{
        char sintomas[100];
        char nome[10];

}cliente;
```

Header file medico.h

Esta header file é semelhante à anterior, sendo que apenas contém duas variáveis relevantes ao funcionamento do programa *medico* (nome, especialidade).

```
typedef struct
{
        char nome[10];
        char  especialidade[15];

}especialista;
```

Header file balcao.h

A header file do *balcao* inclui duas estruturas, sendo a única relevante de referenciar a estrutura cliente, pois foi modificada para receber a informação transmitida pelo classificador (especialidade e prioridade) e a posição atribuída na fila a que pertence. Foi ainda incluída a definição de duas constantes

```
char especialidade[15];
int prioridade;
int lugar; //lugar na fila (1 a MAXFILA)
```

```
#define MAXFILA 5 //número máximo permitido de pessoas por fila
#define MAXESPS 5 //número máximo de especialidades
```