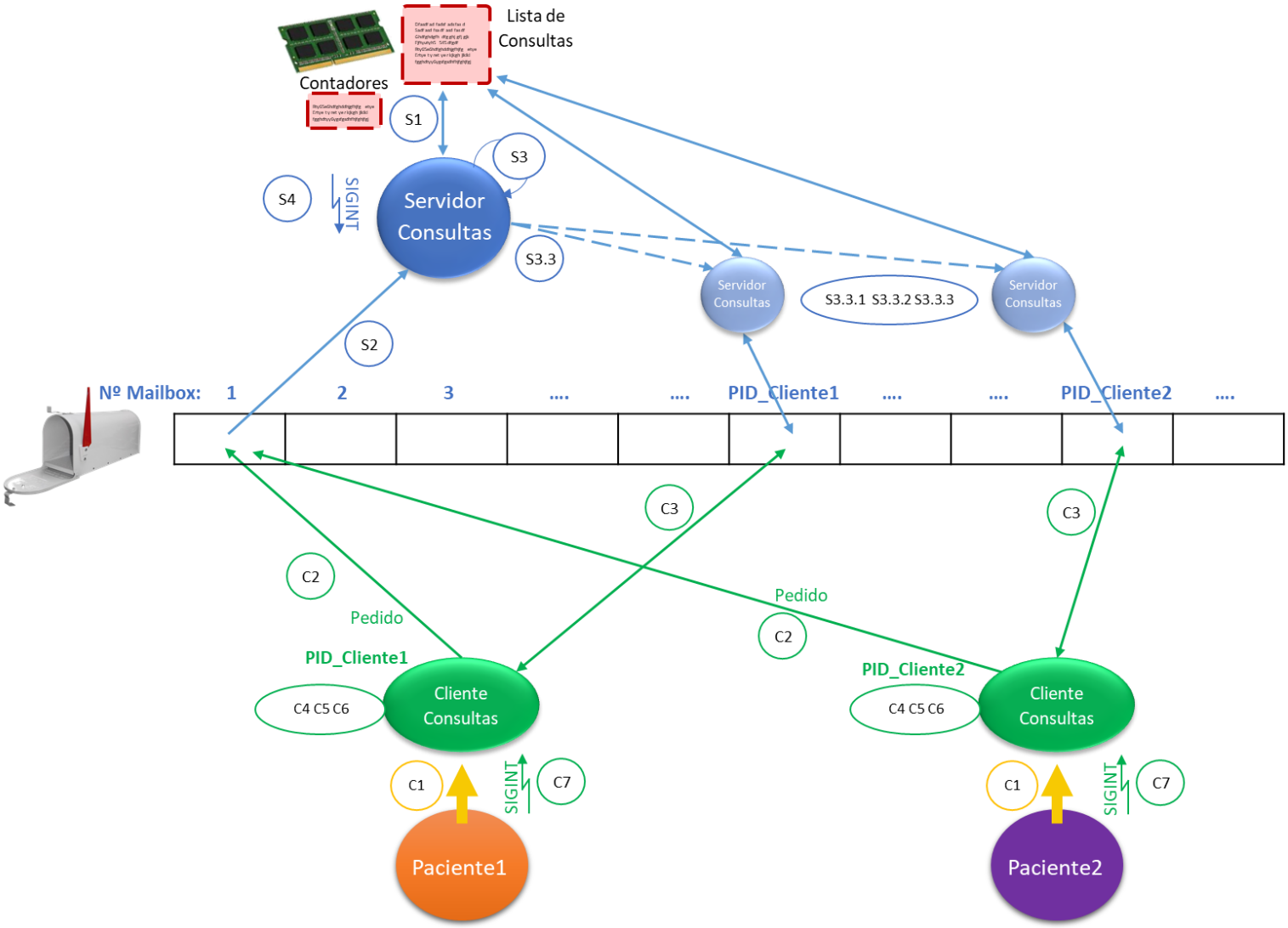
**Projeto *Cliniq-IUL (Parte 3)***

# IPCS

**Data de entrega:** 19 de dezembro de 2020

Nesta parte do trabalho, será implementado um modelo simplificado de Triagem e Consultas de pacientes no sistema Cliniq-IUL, baseado em comunicação por IPCS entre processos, utilizando a linguagem de programação C. Considere o seguinte diagrama, que apresenta uma visão geral da arquitetura pretendida:




**Ideia Geral:** Pretende-se, nesta fase, simular a realização de consultas no sistema Cliniq-IUL. Assim, teremos dois módulos – **Cliente** e **Servidor**.

## Cliente.c

O módulo **Cliente** é responsável pelo encaminhamento dos pacientes. Este módulo será utilizado para solicitar o encaminhamento dos pacientes para as suas consultas. Após esta indicação, o paciente é "encaminhado" e, caso haja disponibilidade, é realizada a consulta, ficando este módulo a aguardar até que a mesma termine.

A estrutura de dados das consultas é a seguinte:

```
typedef struct {
    int tipo;            // Tipo de Consulta: 1-Normal, 2-COVID19, 3-Urgente
    char descricao[100]; // Descrição da Consulta
    int pid_consulta;    // PID do processo que quer fazer a consulta
    int status;          // Estado da consulta: 1-Pedido, 2-Iniciada,
} Consulta;              //              3-Terminada, 4-Recusada, 5-Cancelada
```

Assim, definem-se as seguintes tarefas a desenvolver:

**C1)** Pede ao Paciente (utilizador) que preencha os campos tipo (Tipo de consulta, valores 1, 2 ou 3) e descrição (e.g., "Clínica Geral", "Ortopedia", "Pneumologia").

**C2)** Cria um elemento do tipo **Consulta** com as informações sobre a consulta recolhidas anteriormente, preenchendo o valor pid_consulta com o PID deste processo Cliente. Escreve as informações desse elemento **Consulta** num **1-Pedido** ao Servidor usando a mailbox (ou tipo) 1 da Message Queue;

**C3)** Fica à escuta de mensagens do servidor na Message Queue, na mailbox com o número do seu PID;

**C4)** Se receber uma mensagem com status **2-Iniciada**, escreve no ecrã a mensagem "Consulta iniciada para o processo <pid_consulta>", e aguarda a conclusão da consulta;

**C5)** Se receber uma mensagem com status **3-Terminada**, escreve no ecrã a mensagem "Consulta concluída para o processo <pid_consulta>" e termina o processo. Atenção que apenas deverá aceitar esta mensagem se anteriormente tiver recebido uma mensagem para iniciar a consulta, caso contrário assinala um erro;

**C6)** Se receber uma mensagem com status **4-Recusada**, com a indicação do servidor de que não é possível realizar a consulta. Escreve no ecrã a mensagem "Consulta não é possível para o processo <pid_consulta>" e termina o processo;

**C7)** O módulo Cliente deve armar o sinal **SIGINT**, para que, no caso do Paciente não querer ficar à espera, possa terminar o pedido com o atalho <CTRL+C>. Nesse caso, escreve no ecrã a mensagem "Paciente cancelou o pedido". Envia ao **servidor dedicado** (mailbox com o PID do cliente) uma mensagem com a indicação de consulta **5-Cancelada**, e termina o processo.

# Servidor.c

O módulo **Servidor** de Consultas é responsável pela realização das consultas que chegam ao sistema Cliniq-IUL. Este módulo estará normalmente ativo, à espera de pedidos de consulta. As consultas têm duração de DURACAO segundos (por omissão, 10 segundos). Findo o tempo da consulta, este módulo sinaliza o paciente de que a sua consulta terminou. Este módulo deverá possuir registos em memória partilhada dos contadores de consultas por tipo, e da lista com capacidade para agendar 10 consultas:

```
Consulta lista_consultas[10];
```

O módulo de **Servidor** de Consultas é responsável pelas seguintes tarefas:

**S1)** Se a Memória Partilhada não existir, cria a lista de consultas em Memória Partilhada e inicia essa lista com o campo tipo = –1 ("Limpar" a lista de consultas). Cria também contadores para cada tipo de consultas e para consultas perdidas, iniciados a 0. Escreve no ecrã que iniciou a memória;

**S2)** Fica à escuta de mensagens de clientes na Messsage Queue, na mailbox (tipo) com o número 1;

**S3)** Se receber uma mensagem com status **1-Pedido**:

**S3.1)** Lê a informação da mensagem;

**S3.2)** Escreve no ecrã a mensagem "Chegou novo pedido de consulta do tipo <tipo>, descrição <descricao> e PID <pid_consulta>" com os campos preenchidos corretamente;

**S3.3)** O módulo **Servidor** cria um processo filho (fork) que irá ser o **servidor dedicado** a este cliente. O servidor principal volta a esperar novas mensagens de clientes. Por sua vez, **o servidor dedicado**:

**S3.3.1)** Verifica se a Lista de Consultas tem alguma "vaga". Se todas as entradas da Lista de Consultas estiverem ocupadas, escreve no ecrã "Lista de consultas cheia", manda uma mensagem **4-Recusada** ao processo <pid_consulta>, e incrementa o contador de consultas perdidas;

**S3.3.2)** Se a Lista de Consultas tiver vaga, insere a consulta na lista de consultas, com os campos devidamente preenchidos, escreve no ecrã "Consulta agendada para a sala <índice_da_lista>", e incrementa o contador de consultas do tipo correspondente. Envia uma mensagem **2-Iniciada** ao processo <pid_consulta> correspondente, a indicar o início da consulta. Aguarda DURACAO segundos e escreve no ecrã "Consulta terminada na sala <índice_da_lista>". Envia a mensagem **3-Terminada** e termina o processo servidor dedicado (filho);

**S3.3.3)** Se, durante a DURACAO da consulta (entre as mensagens **2-Iniciada** e a **3-Terminada**), o cliente enviar a mensagem **5-Cancelada** ao servidor dedicado (na mailbox com PID do cliente), o servidor dedicado deverá escrever no ecrã "Consulta cancelada pelo utilizador <pid_consulta>", e termina o seu processo (filho).

**S4)** O módulo **Servidor** de Consultas deve armar e tratar o sinal **SIGINT**, para que possa ser encerrado com o atalho <CTRL+C>, mostrando no ecrã o valor atual das estatísticas e termina o processo **Servidor**. Não precisa guardar os valores das estatísticas nem da lista de consultas em ficheiro, porque esses valores ficam guardados na Memória Partilhada, para quando correr novamente.

Chama-se a atenção dos alunos que a lista de consultas é verificada e atualizada por todos os servidores dedicados, criando novas entradas por cada cliente novo e terminando as mesmas após a realização da consulta. Da mesma forma, os mesmos servidores dedicados são responsáveis por atualizar os contadores das estatísticas de consultas. É necessário acautelar a exclusão no acesso às zonas críticas.

**S5) (extra)** Como se pode ter apercebido, o fato do servidor (pai) não ter previsto fazer wait() aos servidores dedicados (filhos) fará com que os mesmos fiquem num estado especial quando terminarem as suas consultas dedicadas. Desenhe e implemente uma solução para que tal não aconteça, fazendo com que o servidor (pai) ocasionalmente "liberte" os filhos que tem neste estado especial.

Boa Sorte!!!

Os docentes