Trabalho Prático 3
Simulação de Serviço Integrado de Gestão de Alunos (SIGA)

1 Introdução

Este projeto continua a simulação de um sistema de Serviço Integrado de Gestão de Alunos (SIGA) iniciado nos Trabalhos Práticos 1 e 2, onde os estudantes submetem pedidos de ajuda e os agentes de suporte tratam esses pedidos.

Podem usar como base a vossa solução do Trabalho Prático 2 ou a versão partilhada pelos professores depois da data de entrega do trabalho 2. **Será valorizado a utilização da vossa solução do trabalho 2**, com todos os melhoramentos que forem necessários ou considerarem adequados.

Iremos manter os 3 programas desenvolvidos no Trabalho Prático 2, mas serão acrescentadas novas funcionalidades e adicionado um novo programa **admin.c**.

Recapitulando do trabalho 2:

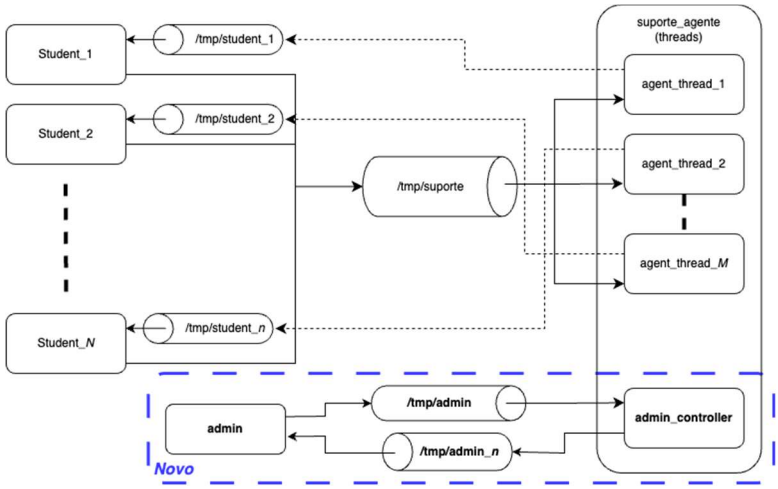
1. **student.c**: Este **programa é feito em C** e é responsável por submeter pedidos ao support_agent
2. **support_agent**: Este **programa é feito em C** e trata os pedidos. Felizmente, temos um agente com várias threads a responder.
3. **support_desk**: Este programa **é feito em (Bash) script** e é responsável por controlar tudo. Ou seja, este programa cria o name pipe, executa os **students** e o **support_agent**.

Novo programa:

4. **admin.c** (novo): Este **programa é feito em C** e interage com o **support_agent**.

1.1 Arquitetura dos processos

A comunicação entre os *student* e o *support_agent* é feita por named pipes e continua igual ao Trabalho Prático 2, com componentes adicionais, marcados a azul na figura abaixo.



Trabalho Prático 3
Simulação de Serviço Integrado de Gestão de Alunos (SIGA)

1.2 student.c

O **student.c** é feito em C e é responsável por submeter pedidos ao support_agent e receber as respetivas respostas. Um pedido é enviado para o named pipe do support_agent (/tmp/supporte); a resposta é recebida no named pipe do student (ex: "/tmp/student_01").

A invocação do programa student é feita com os seguintes parâmetros:

- *Número do processo student*
- *Número de aluno inicial*
- *Número de alunos a inscrever*
- *Nome do pipe para enviar mensagens para o support_agent (opcional)*

Exemplo A: "***./student 1 0 7 /tmp/supportex***"

Executa student com:

- *Número do processo student:* **1**
- *Número de aluno inicial:* **0**
- *Número de alunos a inscrever:* **7**
- *Nome do pipe para enviar os pedidos*: **/tmp/supportex**

Neste trabalho 3, o student envia ao support_agent uma mensagem por cada aluno e disciplina a inscrever, e recebe a respetiva resposta. Considere que **existem 10 disciplinas**; cada aluno tem de ser inscrito em **5 disciplinas** diferentes, **escolhidas aleatoriamente** entre 0 e 9.

Para inscrever um aluno, o student envia para o pipe do support_agent uma mensagem no formato "número_aluno, disciplina, nome_pipe_resposta" e recebe como resultado o horário onde o aluno ficou inscrito. A resposta, recebida no pipe indicado na mensagem, é o número do horário onde o aluno foi inscrito, ou -1 se não havia mais vagas para aquela disciplina. No exemplo acima, o processo student 1 vai por 35 vezes enviar um pedido e receber a resposta, um pedido para cada um dos 7 alunos e 5 disciplinas a inscrever (7 x 5).

Exemplo: o student recebeu como número de aluno inicial "100" e está a inscrever o 2º aluno (aluno nº 101) na disciplina 4

- Pedido: "`101 4 /tmp/student_1`" - esta mensagem vai inscrever o aluno número 101 na disciplina 4 e fica à espera da mensagem de resposta no pipe "/tmp/student_1".
- Resposta: "`2`" – o aluno 101 ficou inscrito na disciplina 4 no horário 2.
- Resposta: "`-1`" – o aluno 101 não ficou inscrito na disciplina 4 pois não havia mais horários disponíveis nessa disciplina

No início, o student deve escrever a seguinte informação no stdout:

"student <nstud>: aluno inicial=<num_aluno_inicial>, número de alunos=<num_alunos>"

Durante a execução do student, para cada estudante inscrito deve escrever o seguinte:

"student <stud_id>, aluno <num_aluno>: dis1/hor, dis2/hor, dis3/hor, dis4/hor, dis5/hor"

Trabalho Prático 3
Simulação de Serviço Integrado de Gestão de Alunos (SIGA)

Exemplo:

- `student 1: aluno inicial=0, número de alunos=7`

- `student 1, aluno 0: 2/0, 5/1, 6/1, 7/1, 8/2`
- `student 1, aluno 1: 1/0, 2/1, 6/1, 7/2, 9/1`
- `student 1, aluno 2: 0/0, 1/1, 3/0, 5/1, 6/2`
  …
- `student 1, aluno 6: 0/0, 1/1, 3/0, 6/-1, 8/-1`
  (para o aluno 6 já não há vagas nas disciplinas 6 e 8)

1.3 admin.c

O **admin.c** é um novo programa, também feito em C, e é responsável por submeter novos pedidos ao support_agent e receber as respetivas respostas. Os pedidos do admin são enviados para um novo pipe do support_agent (/tmp/admin); a resposta é recebida no named pipe do admin (ex: "/tmp/admin_1").

O admin pode enviar ao support_agent 3 tipos de pedidos:

1. **Pedir_horários**: obter as disciplinas e horários em que um aluno está inscrito

Pedido: "<codop>,<num_aluno>,<pipe_resposta>"
codop: código da operação "Pedir_horários" (ex: "1")
num_aluno: número do aluno que se pretende obter as disciplinas e horários
pipe_resposta: nome do pipe onde vai receber a resposta

Resposta: "<num_aluno>,<d/h>,<d/h>,<d/h>,<d/h>,<d/h>
num_aluno: número do aluno que se pretende obter as disciplinas e horários
d: disciplina
h: horário

Exemplo:
Pedido: "`0,2,/tmp/admin_resp`" – pedir as disciplinas e horário do aluno nº 2
Resposta: "`2,0/0,1/1,3/0,5/1,6/2`" – o aluno 2 está inscrito na disciplina 0, horário 0, disciplina 1, horário 1, disciplina 3, horário 0, disciplina 5, horário 1 e disciplina 6, horário 2

2. **Gravar_em_ficheiro**: pedir para o support_agent gravar num ficheiro as disciplinas e horários em que cada aluno está inscrito. O ficheiro deve ter o formato CSV (Comma Separated Values) com uma linha por aluno.

Pedido: "<codop>,<nome_ficheiro>,<pipe_resposta>"
codop: código da operação "Gravar_em_ficheiro" (ex: "2")
<nome_ficheiro>: nome do ficheiro onde deve ser escrita a informação

O ficheiro tem o seguinte formato:

```
#aluno,d0,d1,d2,d3,d4,d5,d6,d7,d8,d9
```

#aluno: número de aluno
di: horário em que o aluno está inscrito na disciplina i

Trabalho Prático 3
Simulação de Serviço Integrado de Gestão de Alunos (SIGA)

Se o aluno não estiver inscrito nessa disciplina, o campo deve ficar vazio.
Resposta: <num_alunos>

num_alunos: número de alunos inscritos, corresponde ao número de linhas escritas no ficheiro. Se houve um erro na escrita do ficheiro, num_alunos deve ser -1.

Exemplo de ficheiro:

```
0,,,0,,,1,1,1,2,,
1,,0,1,,,,1,2,,1
2,0,1,0,,1,2,,,,
```

O aluno 0 está inscrito na disciplina 2, horário 0, disciplina 5, horário 1, disciplina 6, horário 1, disciplina 7, horário 1, disciplina 8, horário 2
O aluno 1 está inscrito na disciplina 1, horário 0, disciplina 2, horário 1, disciplina 6, horário 1, disciplina 7, horário 2, disciplina 9, horário 1
O aluno 2 está inscrito na disciplina 0, horário 0, disciplina 1, horário 1, disciplina 3, horário 0, disciplina 5, horário 1, disciplina 6, horário 2

3. **Terminar**: terminar o support_agent

Pedido: "<codop>,<pipe_resposta>"
codop: código da operação "Terminar" (ex: "3")
pipe_resposta: nome do pipe onde vai receber a resposta
Resposta: "Ok". A resposta é sempre "Ok". A sua receção confirma que o support_agent recebeu corretamente a mensagem e vai terminar ordeiramente

O admin deve ler do teclado os pedidos do utilizador (pedir horários, gravar em ficheiro, terminar), enviar o pedido ao support_agent, receber a resposta e escrever no ecrã o resultado do pedido.

Melhoramentos para obter melhor nota (exemplos):

- Implementar um menu com os pedidos possíveis

1.4 support_agent.c

O **support_agent.c** é **feito em C** e é responsável por receber pedidos dos students, processá-los e enviar-lhes as respostas.

Os pedidos dos processos student são recebidos no named pipe "/tmp/suporte"; a resposta a cada student é enviada para o named pipe desse processo student, cujo nome está indicado na mensagem.

O support_agent.c tem de manter uma estrutura de dados em memória com as várias disciplinas, os vários horários por disciplina, e para cada horário, o número de vagas existente, o número de alunos inscritos e o número de cada aluno inscrito nesse horário.

Este programa tem várias threads que **têm de se sincronizar no acesso às estruturas de dados partilhadas**.

Uma **outra nova funcionalidade** do support_agent é responder aos pedidos do programa **admin**, enviados por um pipe adicional específico, com o nome "/tmp/admin". Ver acima a descrição do

Trabalho Prático 3
Simulação de Serviço Integrado de Gestão de Alunos (SIGA)

admin e o que é esperado que o support_agent faça para cada pedido. O support_agent tem de ter uma **thread dedicada** a receber e processar os pedidos neste pipe "/tmp/admin".

Melhoramentos para obter melhor nota (exemplos):

- Na discussão do Trabalho Prático 3, **executar o support_agent num debugger** (gdb, VSCode, etc), permitindo colocar breakpoints e inspecionar variáveis
- Guardar os alunos inscritos por disciplina e horário **numa lista duplamente ligada**
- Implementar sincronização com read-write locks entre as threads que respondem aos students e a thread que responde ao admin
- Suportar vários admins simultaneamente
- Implementar pedido adicional do admin para mudar o horário de um student
- Implementar pedido adicional do admin para devolver o número de alunos inscritos numa disciplina

1.5 support_desk.sh

O **support_desk** continua a ser feito com um script bash. A sua função é inicializar o sistema, lançar o support_agent e os students e no final remover os pipes criados. O admin deve ser lançado num terminal à parte.

2 Avaliação

A avaliação segue as mesmas regras do trabalho anterior.

3 Submissão e Deadline

A submissão do trabalho é feita da seguinte forma:

1. Fazer commit das alterações e executar o comando push no vosso repositório no GitHub, para o qual convidaram previamente os professores, para que o trabalho fique acessível aos professores
2. Enviar um email aos professores dizendo que o trabalho foi submetido:
    a. Email dos professores: paulo.guedes@ulusofona.pt; daniel.silveira@ulusofona.pt; martijn.kuipers@ulusofona.pt
    b. Subject: "SO - Trabalho 2 submetido, <nome aluno>, <nº aluno>"
    Exemplo: "`Subject: SO - Trabalho submetido, Maria Silva, 12345678`"
    c. O conteudo do email tem de ter um link para o vosso github.
    d. Anexos: os ficheiros .c/.h e .sh

Têm de submeter o vosso trabalho até às **23:59h de domingo dia 15 de dezembro de 2024** (horário de Lisboa).

O trabalho será discutido por cada aluno com o professor no horário da aula prática dessa semana, nos dias 18, 19 e 20 de dezembro de 2024. **A discussão do trabalho é obrigatória.** A nota do trabalho só será atribuída após a discussão.