Ciências ULisboa | Informática

2020/21

Sistemas Operativos

Projeto – Fase 2

1. Introdução

A leitura prévia do enunciado da fase anterior é fundamental e deve ser considerada como introdução a este enunciado. O objetivo geral desta fase é a escrita e leitura de ficheiros, a gestão de tempo, e a captura e tratamento de alarmes e sinais.

2. Descrição Geral

A fase 2 do projeto irá partir do trabalho realizado na fase 1, alterando alguns componentes e adicionando outros novos. Os principais componentes a desenvolver nesta fase são:

- **Ficheiro de Entrada** – a leitura dos argumentos iniciais do ***SOVACCINES*** (ex.: tamanho dos buffers, número de clientes, entre outros) deixará de ser feita pela linha de comandos e passará a ser feita através de um ficheiro de entrada, onde em cada linha é guardado um argumento diferente. O nome deste ficheiro é passado pelos argumentos da linha de comandos (ex., `$./sovaccines config.txt`).
- **Temporização** – os clientes, proxies e servidores passam a registar, nas operações que recebem, o instante temporal em que as processam.
- **Ficheiro de Log** – todas as operações introduzidas pelo utilizador, depois de executadas, passam também a ser guardados num ficheiro de log, que servirá como histórico de utilização do ***SOVACCINES***.
- **Alarmes** – é definido um novo parâmetro de entrada (passado no ficheiro de entrada) que define um intervalo de tempo para um alarme que, quando ativado, imprime para o ecrã o estado atual de todas as operações.
- **Sinais** – é necessário efetuar a captura do sinal de interrupção de programa para efetuar o fecho normal do ***SOVACCINES*** e a libertação de todos os semáforos e zonas de memória.
- **Ficheiro de Estatísticas** – as estatísticas finais do ***SOVACCINES***, para além de incluírem estatísticas de processos (i.e., clientes, proxies e servidores), passam também a incluir estatísticas das operações. Quando o ***SOVACCINES*** termina, estas estatísticas são impressas no ecrã e escritas num ficheiro de estatísticas.

3. Descrição específica

Nesta fase do projeto os alunos terão de desenvolver as suas próprias interfaces (ficheiros .h) e ficheiros de código fonte correspondentes (ficheiros .c), assim como efetuar as alterações que acharem necessárias ao código da fase 1 do seu projeto (excetuando as interfaces) de forma a concretizar os novos objetivos. Nomeadamente, os alunos deverão desenvolver 4 novas interfaces (e ficheiros de código fonte correspondentes): ***configuration.h***, ***log.h***, ***sotime.h*** e ***stats.h***.

3.1. Ficheiro de Entrada

Este ficheiro irá substituir a atual leitura de argumentos da linha de comandos, sendo lido no início da execução do ***SOVACCINES***. Em cada linha deste ficheiro é listado um dos argumentos iniciais de ***SOVACCINES***, seguindo o seguinte formato:

```
max_ops                     //nº máximo de operações
buffers_size                //tamanho dos buffers
n_clients                   //nº de clientes
n_proxies                   //nº de proxies
n_servers                   //nº de servers
log_filename                //nome do ficheiro de log
statistics_filename         //nome do ficheiro de estatísticas
alarm_time                  //temporização para o alarme
```

Atenção aos novos argumentos `log_filename`, `statistics_filename`, e `alarm_time`. O nome do ficheiro de entrada é passado ao ***SOVACCINES*** através dos argumentos da linha de comandos, e passa a ser o único argumento introduzido dessa forma.

Este módulo deve ser desenvolvido no ficheiro ***configuration.h***, sendo também necessário efetuar a ligação entre este e os módulos da fase 1 do projeto.

3.2. Temporização

O módulo de temporização é outro dos novos módulos a desenvolver. Nomeadamente, os clientes, proxies e servidores passam a usar funções de temporização para registar o instante no tempo em que processam cada operação recebida. Adicionalmente, a `main` regista o instante em que uma operação é inicializada (i.e., quando o utilizador pede a sua criação) e o cliente regista quando ela termina (i.e., quando a resposta é recebida pelo cliente).

Para guardar estes tempos, devem ser adicionados novos campos na estrutura *operation*:

```
struct operation {
      ...                                   //manter os campos da 1ª fase
      struct timespec start_time;           //quando a op foi iniciada
      struct timespec client_time;          //quando o cliente processou a op
      struct timespec proxy_time;           //quando o proxy processou a op
      struct timespec server_time;          //quando o server processou a op
      struct timespec end_time;             //quando a op foi concluída
};
```

O registo de tempos deve ser feito usando a função *clock_gettime* da biblioteca *time.h*.

Este módulo deve ser desenvolvido no ficheiro ***sotime.h***, fazendo também as alterações e ligações necessárias com os módulos da fase 1 do projeto.

3.3. Ficheiro de Log

A `main` (processo principal) passa a guardar um registo de todas as instruções executadas pelo utilizador. Tal registo é guardado num ficheiro de log, cujo nome é passado como um dos argumentos do ficheiro de entrada (argumento `log_filename`). As instruções do utilizador (*op*, *read* e *close*) são guardados no seguinte formato:

```
time instruction argument
```

onde `time` é o instante em que a instrução foi feita pelo utilizador, indicando o ano, mês, dia, hora, minuto, segundo e milissegundo. Para indicarem estes tempos podem usar as funções *clock_gettime* e *localtime* da biblioteca *time.h;* `instruction` é a instrução (*op, read ou close*); e `argument` é o argumento da instrução (caso exista) passado pelo utilizador. Segue um exemplo de ficheiro de log:

```
2021-2-31 14:53:19.943 op
2021-2-31 14:53:30.572 read 0
2021-2-31 14:54:21.326 close
```

Este módulo deve ser desenvolvido no ficheiro ***log.h,*** sendo também necessário efetuar as devidas alterações e ligações com os módulos da fase 1 do projeto.

3.4. Alarmes

Neste módulo, pretende-se armar um alarme que em certos intervalos de tempo (ex: X em X segundos) verifique e escreva para o ecrã o estado atual de todas as operações, incluindo: (*i*) as que já foram executadas e processadas e (*ii*) as que ainda não foram invocadas pelo utilizador. O valor desse intervalo de tempo é definido no ficheiro de entrada, correspondendo ao intervalo `alarm_time`. O estado das operações será escrito segundo o seguinte formato:

```
op status start_time client_id client_time proxy_id proxy_time server_id
server_time end_time
```

Aqui, os tempos devem ser apresentados em segundos e no formato *raw*, ou seja, sem aplicar a função *localtime*. Uma operação que ainda não foi executada é reportada com status 0.

Por exemplo, para um *max_ops* = 3, o output do alarme poderia ser:

```
op:0 status:S start: 1617202725 client:1 client_time: 1617202726 proxy:1
proxy_time: 1617202727 server:1 server_time: 1617202728 end: 1617202729
op:1 status:0
op:2 status:0
```

Este módulo deve ser desenvolvido no ficheiro ***sosignal.h***, fazendo também as alterações e ligações necessárias com os módulos da fase 1 do projeto.

3.5. Sinais

Neste módulo, pretende-se capturar o sinal de interrupção de programa (SIGINT – ativado pela combinação de teclas CTRL-C) de forma a libertar todos os recursos do ***SOVACCINES*** e efetuar um fecho normal do programa. Assim, ao capturar este sinal, deve-se proceder à invocação da função *stop_execution* (para tal, os alunos poderão ter que passar as variáveis *main_data*, *communication_buffers* e *semaphores* para variáveis globais).

Este módulo deve ser desenvolvido no ficheiro ***sosignal.h***, fazendo também as alterações e ligações necessárias com os módulos da fase 1 do projeto.

3.6. Ficheiro de estatísticas

Por fim, as estatísticas finais do ***SOVACCINES*** passam a ser escritas para um ficheiro de estatísticas, que irá incluir tanto as estatísticas dos vários processos como das várias operações. O nome do ficheiro de estatísticas é passado como um dos argumentos do ficheiro de entrada (argumento `statistics_filename`).

Este ficheiro terá o seguinte formato:

```
Process Statistics:
      Client … received … requests!
      ...
      Proxy … forwarded … requests!
      ...
      Server … responded to … requests!
      ...

Operation Statistics:
OP: …
Status: …
Client_id: …
Proxy_id: …
Server_id: …
Created: …
Client_time: …
Proxy_time: …
Server_time: …
Ended: …
Total Time: …

OP: …
...
```

Os tempos `created, cliente_time, proxy_time`, e `server_time` indicam o ano, mês, dia, hora, minuto, segundo e milissegundos. Para indicarem estes tempos podem usar a função *localtime* da biblioteca *time.h.*

O tempo `Total Time` é a diferença entre o `end_time` e o `start_time`, sendo apresentado em segundos e milissegundos.

Segue um exemplo concreto:

```
Process Statistics:
      Client 0 received 1 requests!
      Proxy 0 forwarded 1 requests!
      Server 0 responded to 1 requests!

Operation Statistics:
OP: 0
Status: S
Client_id: 0
Proxy_id: 0
Server_id: 0
Created: 2021-2-31 14:53:19.806
Client_time: 2021-2-31 14:53:20.234
Proxy_time: 2021-2-31 14:53:21.575
Server_time: 2021-2-31 14:53:22.143
Ended: 2021-2-31 14:53:23.372
Total Time: 3.566
```

Este módulo deve ser desenvolvido no ficheiro ***stats.h***, sendo também necessário fazer as devidas alterações e ligações com os módulos da fase 1 do projeto.

4. Entrega

A entrega da fase 2 do projeto tem de ser feita de acordo com as seguintes regras:

1. Colocar todos os ficheiros do projeto, de acordo com a estrutura usada na fase 1 do projeto, **bem como um ficheiro README** onde os alunos podem incluir informações que julguem necessárias (ex., nome e número dos alunos que o desenvolveram, limitações na implementação do projeto, etc.), num ficheiro comprimido no formato ZIP. O nome do ficheiro será **`grupoXX-projeto2.zip`** (XX é o número do grupo).
2. Submeter o ficheiro **`grupoXX-projeto2.zip`** na página da disciplina no moodle da FCUL, utilizando a atividade disponibilizada para tal. Apenas um dos elementos do grupo deve submeter, evitando dessa forma que seja escolhida aleatoriamente uma submissão no caso de existirem várias.

Na entrega do trabalho, é ainda necessário ter em conta que:

- **(1) se não for incluído um `Makefile` e (2) se o mesmo não compilar os ficheiros fonte, ou (3) se houver erros de compilação (isto é, se não forem criados os ficheiros objeto e executáveis), o trabalho é considerado nulo**.
- Todos os ficheiros entregues devem começar com um cabeçalho com três ou quatro linhas de comentários indicando o número do grupo, o nome e número dos seus elementos.
- Os programas são testados no ambiente Linux instalado nos laboratórios de aulas, pelo que se recomenda que os alunos desenvolvam e testem os seus programas nesse ambiente. A imagem Linux instalada nos laboratórios pode ser descarregada de https://admin.di.fc.ul.pt/importacao-da-imagem-para-virtualbox-tutorial/

O prazo de entrega é dia 02/05/2021 até às 23h55min.

Após esta data, a submissão do trabalho através do Moodle deixará de ser permitida.