



UNIVERSIDADE DO MINHO

DEPARTAMENTO DE INFORMÁTICA

**Controlo e Monitorização de Processos
e Comunicação**
Trabalho Prático
Sistemas Operativos
Grupo 22

Carlos Gomes (a77185) Márcia Teixeira (a80943)
Rui Armada (a90468)

15 de Junho de 2020

Resumo

O presente documento descreve o trabalho prático realizado no âmbito da disciplina de *Sistemas Operativos* (SO), ao longo do segundo semestre, do segundo ano, do Mestrado Integrado em Engenharia Informática da Universidade do Minho.

O objetivo deste projeto foi desenvolver um serviço de monitorização de execução e de comunicação entre processos que permita a um utilizador fazer a submissão de sucessivas tarefas, tarefas das quais serão enviadas para um servidor que as vai executar e apresentar várias funcionalidades tais como: Se a tarefa foi efetuada com sucesso, mostrar o histórico de todas as tarefas feitas, etc.

Neste relatório iremos descrever com clareza todo o raciocínio e decisões tomadas ao longo da elaboração deste projeto.

Conteúdo

1	Introdução	3
1.1	Objetivo	3
2	Conceção da Solução	3
3	Implementação	4
3.1	Cliente -> Servidor	4
4	Funcionalidades	4
4.1	Execução de Tarefas	4
4.2	Mostrar o resultado da execução de tarefas	4
4.3	Definição de tempo de execução	5
4.4	Mostrar que só pode executar um programa em n segundos	5
4.5	Ajuda	5
4.6	Histórico de tarefas	6
4.7	Terminar tarefas em execução	6
5	Conclusão	7

1 Introdução

O objetivo deste projeto é implementar um serviço de gestão de processos de forma a ser capaz de monitorizar a execução e comunicação entre processos. Para implementarmos este serviço, iremos implementar uma arquitetura Cliente, Servidor, com recurso a pipes e aplicar outros conhecimentos acumulados ao longo da unidade curricular de Sistema Operativos.

1.1 Objetivo

Como foi referido anteriormente, um dos objetivos deste serviço é a monitorização de execução e comunicação entre processos, onde o serviço deverá permitir a um utilizador a submissão de tarefas em que cada uma delas será uma sequência de comandos encadeados por pipes. Para além de iniciar as tarefas, o serviço também deverá ser capaz de identificar quais são as tarefas em execução, assim como terminar essas mesmas. Também deverá oferecer uma interface ‘amigável’ para o utilizador, onde o utilizador irá executar os comandos e visualizar a execução dessas tarefas.

2 Conceção da Solução

No sentido de resolver a problemática apresentada foi necessário partir do princípio que o problema inicial teria de ser dividido em problemas mais pequenos.

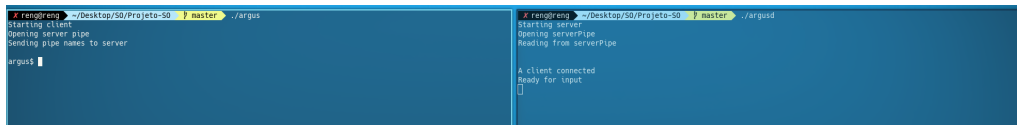
Numa primeira fase foi idealizada a estrutura que viria a ter tanto o servidor como o cliente. Para tal, e sabendo que iriam comunicar através de pipes (com nome), foi tomada a decisão que o servidor apenas teria controlo sobre um pipe inicialmente criado para assegurar que o processo de conexão com um cliente decorreria normalmente. A função deste pipe será a transmissão dos nomes dos pipes, posteriormente utilizados para a comunicação dos pedidos do cliente, ao servidor e consequente resposta aos mesmos. No sentido de evitar colisões nos pipes utilizados para envio de mensagens entre cliente e servidor, cada cliente que efetua uma nova conexão com o servidor cria os seus próprios pipes de leitura e de escrita, que diferem de todos os outros clientes à exceção do sufixo “-in” e “-out”, respetivamente. Assim garante-se que dois clientes distintos não lêem informações um do outro, permitindo que vários estejam ativos em simultâneo, apesar de se poderem encontrar bloqueados à espera de serem atendidos pelo servidor.

Tendo ficado tratado o ponto indispensável do presente projeto, que é a comunicação, passou-se então a uma segunda fase, que seria o desenvolvimento das funcionalidades enunciadas para o funcionamento mínimo do sistema. Já fazendo referência ao cliente singular, foi desenvolvido apenas um sistema básico de leitura de texto e envio para o servidor, que por sua vez tomará conta dos pedidos e é responsável pela resposta de acordo com a usabilidade do input recebido. Por sua vez, o servidor desenvolvido trata-se do sistema mais complexo e portanto mais ponderado do projeto, pelas várias funções que desempenha no mesmo.

3 Implementação

3.1 Cliente -> Servidor

Para o desenvolvimento do serviço, é crucial haver uma comunicação entre o servidor e o cliente. O utilizador poderá agir sobre o servidor através dos argumentos passados na linha de comando do cliente e o servidor deverá receber esses argumentos e executar de acordo com o desejo do cliente.



```
X reng@reng ~/Desktop/S0/Projeto-S0 master ./argus
Starting client
Opening server pipe
Sending pipe names to server
argus$

X reng@reng ~/Desktop/S0/Projeto-S0 master ./argusd
Starting server
Opening serverPipe
Reading from serverPipe
A client connected
Ready for input
```

Figura 1: Cliente->Servidor

4 Funcionalidades

Como já foi referido anteriormente, é pretendido o desenvolvimento de um serviço de monitorização de execução e comunicação de processos. Então para isto acontecer, é necessário a existência de algumas funcionalidades que iremos explorar.

4.1 Execução de Tarefas

Funcionalidade que executa uma tarefa.



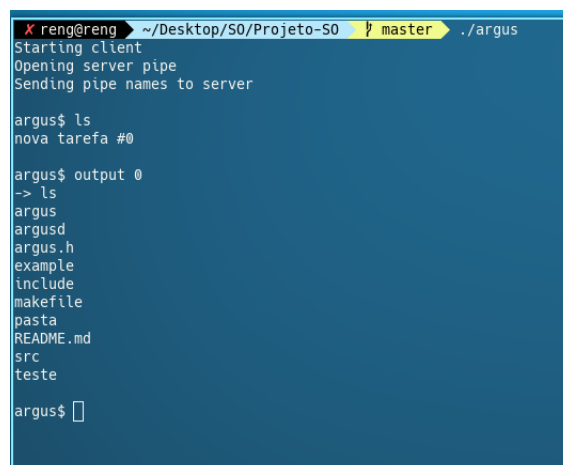
```
X reng@reng ~/Desktop/S0/Projeto-S0 master ./argus
Starting client
Opening server pipe
Sending pipe names to server
argus$ ls
nova tarefa #0
argus$

X reng@reng ~/Desktop/S0/Projeto-S0 master ./argusd
Starting server
Opening serverPipe
Reading from serverPipe
A client connected
Ready for input
-> Task 0 executed successfully
```

Figura 2: Execução de Tarefas

4.2 Mostrar o resultado da execução de tarefas

Funcionalidade para consultar o *standard output* produzido por uma tarefa já executada.



```
X reng@reng ~/Desktop/S0/Projeto-S0 master ./argus
Starting client
Opening server pipe
Sending pipe names to server
argus$ ls
nova tarefa #0
argus$ output 0
-> ls
argus
argusd
argus.h
example
include
makefile
pasta
README.md
src
teste
argus$
```

Figura 3: Mostrar o resultado da execução de tarefas

4.3 Definição de tempo de execução

Funcionalidade que fornece o tempo máximo (segundos) de execução de uma tarefa.

```
argus$ tempo-execucao 10
Got it, thanks!

argus$
```

Figura 4: Definição de tempo de execução

4.4 Mostrar que só pode executar um programa em n segundos

Funcionalidade que obriga a execução de um programa num dado periodo de tempo definido pelo Utilizador.

```
X reng@reng ~/Desktop/S0/Projeto-S0 master ./argus
Starting client
Opening server pipe
Sending pipe names to server

argus$ ls
nova tarefa #0

argus$ output 0
-> ls
argus
argusd
argus.h
example
include
makefile
pasta
README.md
src
teste

argus$ tempo-execucao 10
Got it, thanks!

argus$ sleep 30
nova tarefa #1

X reng@reng ~/Desktop/S0/Projeto-S0 master ./argusd
Starting server
Opening serverPipe
Reading from serverPipe

A client connected
Ready for input

-> Task 0 executed successfully

-> Task 1 timed out (execution time exceeded)
```

Figura 5: Mostrar que só pode executar um programa em n segundos

4.5 Ajuda

Funcionalidade que apresenta o menu de ajuda para a Utilização da Aplicação.

```
argus$ ajuda

----->>> Here's a list of the available commands! <<<<-----

-i <argumento1>, tempo-inatividade <argumento1>
-> Define o tempo máximo, em segundos, de inatividade de comunicação num pipe anónimo

-m <argumento1>, tempo-execucao <argumento1>
-> Define o tempo máximo (segundos) de execução de uma tarefa

-e <p1|p2|...|pn>, executar <p1|...|pn>
-> Executa uma tarefa da linha de comandos

-l, listar
-> Lista as tarefas em execução

-t n, terminar n
-> Termina uma tarefa em execução

-r, historico
-> Histórico de tarefas terminadas
```

Figura 6: Ajuda

4.6 Histórico de tarefas

Funcionalidade que lista registro historico de tarefas terminadas.

```
argus$ historico
#0, Finished: -> ls
#1, Max execution: -> sleep 30
#2, Finished: -> sleep 30
#3, Finished: -> sleep 30
argus$
```

Figura 7: Histórico de tarefas

4.7 Terminar tarefas em execução

Funcionalidade que termina uma tarefa em execução.

```
argus$ listar
#2: sleep 30

argus$ sleep 30
nova tarefa #3

argus$ terminar 3
Task terminated successfully
argus$
```

Figura 8: Terminar tarefas em execução

5 Conclusão

Face ao problema apresentado e analisando criticamente a solução proposta concluimos que cumprimos as tarefas, conseguindo atingir os objetivos definidos. De facto, foram implementadas todas as funcionalidades apresentadas, quer básicas quer avançadas. Deste modo, foi então construído um sistema de Controlo e Monitorização de Processos e Comunicação perfeitamente funcional, capaz de executar com exatidão todas as tarefas propostas pelo enunciado deste trabalho prático.