

**Sistemas Operativos**

**Estatística de Processamento em Bash**

**Trabalho 1**

**Professor:**

Nuno Lau ([nunolau@ua.pt](mailto:nunolau@ua.pt))

**Pedro** **Sobral**, 98491, P3

**Daniel Figueiredo**, 98498, P5

**As distribuições de percentagem são iguais para os membros do grupo**

07/12/2020

Índice

[Introdução 2](#_Toc57677521)

[Conclusão 4](#_Toc57677522)

[Bibliografia 5](#_Toc57677523)

# Introdução

No contexto da disciplina de Sistemas Operativos, foi-nos apresentado a realização deste trabalho prático, que consiste na visualização e no tratamento dos processos que estão a *correr* na nossa máquina. O trabalho prático consiste em programar um script *procstat.sh*, de modo a que este permita visualizar a quantidade de memória total, de memória residente em memória física, o número total de bytes I/O, e a taxa de leitura/escrita (bytes/sec) dos processos selecionados nos últimos *sec* segundos.

A implementação do código bash será feita através do IDE *VSCode*, pois é um editor com que ambos os membros grupo já estão familiarizados derivados a unidades curriculares e projetos passados. A execução de todo o trabalho foi suportada através de um [repositório no GitHub](https://github.com/TheScorpoi/SO_Trabalho01)1, o que facilita em muito todo o *workflow* da realização do mesmo.

O *script procstat.sh* será feito com base na matéria dada nas aulas práticas, onde foram abordados os conceitos base de programação em bash, que nos permitem listar os processos que se encontram a *correr* no nosso computador. De acordo com o guião teremos de executar vários comandos que irão permitir fazer a filtragem da informação. A nossa ideia inicial para guardar a informação passará por utilizar arrays associativos, visto são estruturas de dados onde é possível identificar cada elemento por uma *key,* e por um *value,* de um modo geral é um pouco como os *HashMaps* em *Java*, por exemplo. Depois de ter a informação dos processos toda tratada/formatada e devidamente guardada, iremos criar as *expressões condicionais* para tratar a informação de acordo com as opções passadas no terminal, quando se corre o programa.

Com a realização deste trabalho prático, esperamos veemente alargar os nossos horizontes no que toca a programar em bash, e perceber um pouco mais do funcionamento dos processos do sistema operativo (*Linux - Ubuntu*), visto que é bastante importante para esta unidade curricular.

1 – O repositório encontra-se privado, pelo menos até ao dia da entrega do trabalho (07/12/2020)

A função *listarProcessos* é uma das principais funções do *script*, sendo que é aqui que se lê as informações dos processos e se trata desses dados de forma primordial.

Começamos por *printar* o cabeçalho da nossa tabela, visto que este será sempre impresso, e só queremos que este seja *printado* uma vez a cada execução do *script.* Seguidamente iniciamos um ciclo *for*, que nos irá permitir percorrer todos os processos que estão ativos no nosso computador, cada processo fica na variável *entry* a cada iteração do ciclo for, este ciclo itera sobre o CAMINHO? DIRETORIA? */proc/[[:digit:]]\*,* onde esta último expressão nos irá dar todos os processo processos em que as diretorias são dígitos. Ir ao terminal e correr o cd /proc ou la como é, e mostrar as diretorias so com os números

Agora seguem-se algumas condições para o tratamento dos dados, várias expressões condicionais. O primeiro if [[ -r ????????? ]], serve para, caso não tenhamos permissões de acesso a uma determinada informação do processo, não o vamos listar. Segue-se mais um if \*\*\*\*\*\*\*\*\*\*, neste if é decidido logicamente se o processo possui um comm (O QUE É QUE O -F FAZ??).

SECALHAR POR TITULO AQUI TIPO, IR BUSCAR AS COISAS AO STATUS

Agora chegamos a uma fase do código em que iremos guarda em variáveis locais, alguns atributos necessários para a realização da tabela proposta no guião. Começamos pelo PID, o PID é o identificador do processo em questão, sendo este um valor muito importante pois será usado no procedimento de criação de outra variáveis, p.e. user, startDate.

Usamos o cat /$entry/status, pois é nesse *path* que o PID, se encontra, usamos um *grep -w Pid*, para que devolve-se só a linha que possuía o PID, e um *tr -dc ‘0-9’*, para devolver somente a parte numérica do PID.

Para atribuirmos à variável *user*, o utilizador do processo, fizemos o comando ps, e com o PID, obtivemos o utilizador. Para a data de início do processo, o mecanismo foi o mesmo, com o PID, e o comando ps, guardados a data inicial do processo na variável *startDate*.

Em seguida fomos buscar o VmSize, e o VmRss, que correspondem á memoria FISICA E VIRTUAL??????\*\*\*\*\*, respetivamente, o processo de filtragem da informação foi semelhante ao do PID, usamos um *grep* e um *tr* para que só fossem devolvidos os valores dessas 2 variáveis.

SECALHAR POR TITULO, IR BUSCAR AS COISAS AO IO

Fazemos o if [[ -f $entry/status ]] QUE JÁ NS OQ FAZ\*\*\*\*\*?????

Guardamos o argumento passado á função *listarProcessos*, $1, na variável sec, que são os segundos em que o *script* irá executar um sleep. Em seguida vamos filtrar a informação relativamente ao *rchar* e ao *wchar*, que são o bytes que o processo leu (*rchar1*) e escreveu (wchar1), respetivamente. Fazemos um sleep de sec segundos, e voltamos a ler os valores de *rchar* e *wchar, e guardamos nas variáveis rchar2 e wchar2*. De seguida, vamos calcular o *rateR, e o rateW*, que são a taxa de leitura e escrita num determinado intervalo de tempo, neste caso, no intervalo *sec*. Em ambos o cálculo é simples, uma subtração do último valor lido com o valor lido primeiro, a dividir pelo número de segundos em que o *sleep* esteve ativo. Este calculo é feito tanto para o rateR como para o rateW.

Em seguida criamos a variável *comm*, onde irá ficar guardado o nome do processo em analise, fizemos a substituição do espaço por um ‘\_’, pois durante a realização do trabalho prático havia alguns processos em que o *comm*, eram duas palavras, sendo que depois na hora de ordenar, a tabela não ordenava corretamente, derivado a isso, desta forma o problema ficou instantaneamente resolvido.

# Conclusão

Ao longo da realização deste trabalho

# Bibliografia

<https://man.cx/bash>

<https://stackoverflow.com/>

<https://www.cyberciti.biz/faq/linux-unix-formatting-dates-for-display/>

<https://unix.stackexchange.com/questions/176198/determine-the-owner-of-the-session-of-a-process>

<https://www.linuxforce.com.br/comandos-linux/comandos-linux-comando-printf/>

- Para a realização deste trabalho prático também consultamos os slides disponibilizados pelo Professor na página do *e-learning* da unidade curricular Sistemas Operativos.