

# Trabalho 1

## Estatísticas de processos em **bash**

### Guião

O objectivo do trabalho é o desenvolvimento de um *script* em **bash** que apresenta estatísticas sobre a memória usada por processos e sobre a quantidade de I/O que uma selecção de processos estão a efetuar. Este *script* permite visualizar a quantidade de memória total de um processo, a quantidade de memória física ocupada por um processo, o número total de bytes de I/O que um processo leu/escreveu e também a taxa de leitura/escrita correspondente aos últimos *s* segundos para de um processo (o valor de *s* é passado como parametro).

O script **procstat.sh** permite a visualização da quantidade de memória total e da memória residente em memória física (linhas **VmSize** e **VmRSS** de **/proc/[pid]/status**), do número de total de bytes de I/O (linhas **rchar** e **wchar** de **/proc/[pid]/io**) e da taxa de leitura/escrita (em bytes por segundo) dos processos seleccionados nos últimos *s* segundos (calculadas a partir de 2 leituras de **/proc/[pid]/io** com intervalo de *s* segundos). Este *script* tem um parâmetro obrigatório que é o número de segundos que serão usados para calcular as taxas de I/O. A selecção dos processos a visualizar pode ser realizada através de uma expressão regular que é verificada com o comando (tal como aparece em **/proc/[pid]/comm**) associado (opção **-c**), ou através da definição de um período temporal para o início do processo. A especificação do período temporal faz-se através da data mínima (opção **-s**) e data máxima (opção **-e**) para o início do processo. A selecção dos processos pode ainda ser realizada através do nome do utilizador (opção **-u**). A visualização está formatada como uma tabela, com um cabeçalho, aparecendo os processos por ordem alfabética. O número de processos a visualizar é controlado pela opção **-p**. Existem opções para alterar a ordenação da tabela (**-m** – sort on **MEM**↑, **-t** – sort on **RSS**↑, **-d** – sort on **RATER**↑, **-w** – sort on **RATEW**↑ e **-r** – reverse).

Seguem-se exemplos do que pode aparecer na consola durante a execução deste *script*:

```
$ ./procstat.sh 10
```

COMM	USER	PID	MEM	RSS	READB	WRITEB	RATER	RATEW	DATE
bash	nlau	10174	3724656	452880	24690	58597	3770687.30	14239.00	Sep 12 11:45
dice	sop0200	5036	123770	45005	2914000	1356	4022.70	5114.70	Sep 23 18:14
dropbox	sop0100	2636	3288072	335932	918784	1356760	40022.70	31114.50	Sep 19 08:49

```
$ ./procstat.sh -c "d.*" 10
```

COMM	USER	PID	MEM	RSS	READB	WRITEB	RATER	RATEW	DATE
dice	sop0200	5036	123770	45005	2914000	1356	4022.70	5114.70	Sep 23 18:14
dropbox	sop0100	2636	3288072	335932	918784	1356760	40022.70	31114.50	Sep 19 08:49

```
$ ./procstat.sh -u sop0100 10
```

COMM	USER	PID	MEM	RSS	READB	WRITEB	RATER	RATEW	DATE
dropbox	sop0100	2636	3288072	335932	918784	1356760	40022.70	31114.50	Sep 19 08:49

```
$ ./procstat.sh -s "Sep 10 10:00" -e "Sep 20 18:00" 10
```

COMM	USER	PID	MEM	RSS	READB	WRITEB	RATER	RATEW	DATE
bash	nlau	10174	3724656	452880	1532469	58597	3770687.30	14239.00	Sep 12 11:45
dropbox	sop0100	2636	3288072	335932	918784	1356760	40022.70	31114.50	Sep 19 08:49

```
$ ./procstat.sh -t -c "d.*" 10
```

COMM	USER	PID	MEM	RSS	READB	WRITEB	RATER	RATEW	DATE
dropbox	sop0100	2636	3288072	335932	918784	1356760	40022.70	31114.50	Sep 19 08:49
dice	sop0200	5036	123770	45005	2914000	1356	4022.70	5114.70	Sep 23 18:14

```
$ ./procstat.sh -w -c "d.*" 10
```

COMM	USER	PID	MEM	RSS	READB	WRITEB	RATER	RATEW	DATE
dropbox	sop0100	2636	3288072	335932	91878	1356760	40022.70	31114.50	Sep 19 08:49
dice	sop0200	5036	123770	45005	2914000	1356	4022.70	5114.70	Sep 23 18:14

```
$ ./procstat.sh -m -r -c "d.*" 10
```

COMM	USER	PID	MEM	RSS	READB	WRITEB	RATER	RATEW	DATE
dice	sop0200	5036	123770	45005	2914000	1356	4022.70	5114.70	Sep 23 18:14
dropbox	sop0100	2636	3288072	335932	918784	1356760	40022.70	31114.50	Sep 19 08:49

```
$ ./procstat.sh -m -r -c "d.*" -p 1 10
```

COMM	USER	PID	MEM	RSS	READB	WRITEB	RATER	RATEW	DATE
dice	sop0200	5036	123770	45005	2914000	1356	4022.70	5114.70	Sep 23 18:14

A estrutura da linha de comando dos *scripts* deve ser sempre validada, garantindo assim que os parâmetros que foram usados estão de acordo com o esperado.

A execução do trabalho poderá ser suportada através de um repositório GIT ou SVN a criar na plataforma **code.ua.pt**.

O trabalho será realizado em grupos de 2 alunos ou individualmente. Durante a execução do trabalho deve ser respeitado um exigente código de ética que impede o plágio, sob qualquer forma, bem como o desenvolvimento do trabalho por elementos externos ao grupo ou a partilha de código entre grupos distintos.

A entrega do trabalho será realizada através do **elearning.ua.pt** e deverá incluir o código fonte da solução encontrada e um relatório que descreve qual a abordagem usada para resolver o problema e os testes realizados para validar a solução.

Dicas: alguns comandos que poderão ser úteis para este trabalho são **awk**, **bc**, **cat**, **cut**, **date**, **getopts**, **grep**, **head**, **ls**, **printf**, **sleep**, **sort**

**Data de entrega do trabalho: 7 de dezembro de 2020**