

Instituto Politécnico Cávado do Ave
Curso engenharia de Sistemas Informáticos
Sistemas Operativos e Sistemas Distribuídos

Autores

Carlos Santos Nº 19432

João Rodrigues Nº 19431

João Ricardo Nº 18845

Trabalho Prático

Data: 29/12/2020

ÍNDICE

1. Parte 1	3
2. Parte 2	5
3. Parte 3	8
4. Conclusão	16
5. Execução das Tarefas	17

Parte 1

Implementação de um conjunto de comandos para manipulação de ficheiros

Era pretendido que se implementa-se os seguintes comandos:

1. **mostra** ficheiro - Este comando deve mostrar no ecrã o conteúdo do ficheiro indicado.
2. **conta** ficheiro - Este comando deve contar o número de caracteres existentes de um ficheiro.
3. **apaga** ficheiro - Este comando deve apagar o ficheiro com o nome indicado.
4. **informa** ficheiro - Este comando apresenta a informação do sistema de ficheiros em relação ao ficheiro dado.
5. **acrescenta** origem destino - Este comando deve acrescentar o conteúdo da "origem" no final do "destino".
6. **lista** [caminho] - Este comando deve apresentar uma lista de todas as pastas e ficheiros existentes no caminho indicado ou na diretoria atual se não especificado.

Através de chamada de funções ao sistema (*system calls*). Estes comandos implementados em C serão invocados através de um interpretador de comandos.

```
int main(int argc, char* argv){
    int fd;

    // Verifica se foi enviado o segundo argumento: comando + ficheiro
    if (argc != 2) {
        fputs("Erro na apresentacao do ficheiro.\n", stderr);
        exit(EXIT_FAILURE);
    }

    // Executa a funcao system call, unlink
    fd = unlink(argv[1]);

    // Verifica se existe um erro ao executar a funcao
    if (fd < 0) {
        perror("Erro ao apagar o ficheiro");
        exit(EXIT_FAILURE);
    }

    // Fecha o ficheiro
    close(fd);

    exit(EXIT_SUCCESS);
}
```

Figura 1: Função Apaga

Tanto nesta função como nas outras, o programa recebe como argumento o ficheiro ou diretoria em questão verificando sempre se realmente recebeu corretamente os argumentos requisitados. De seguida executa a função system call e verifica com o valor devolvido pela função se ocorreu algum erro. Para finalizar o ficheiro deve ser fechado com a função close.

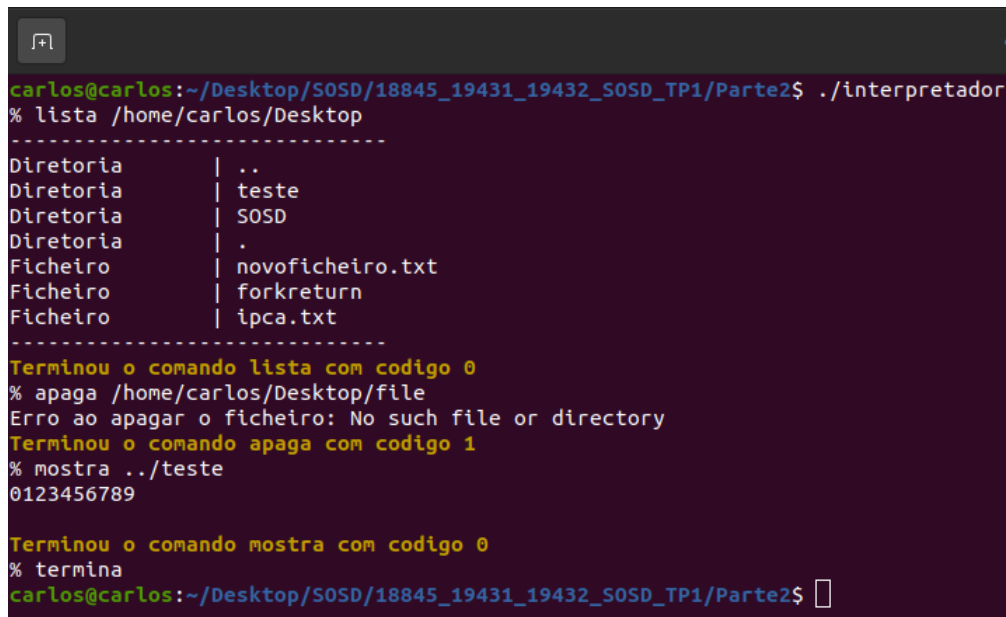
¹

¹<https://linux.die.net/man/>

Parte 2

Implementação de um interpretador de comandos

No sentido de substituir o interpretador de comandos habitual, por um novo interpretador personalizado, era importante implementar um programa que execute o comando através de primitivas de execução genérica de processos. Cada comando deverá dar origem a um novo processo e adicionalmente, poder considerar que a execução do interpretador deve ser suspensa até ao comando indicado estar concluído. O interpretador deverá indicar sempre se o comando conclui com ou sem sucesso, através do seu código de terminação/erro. O programa deverá permitir executar vários comandos sequencialmente, isto é, um a seguir ao outro, até o utilizador indicar o comando especial “termina” que termina esta aplicação.

A terminal window with a dark background and light-colored text. The prompt is 'carlos@carlos:~/Desktop/SOSD/18845_19431_19432_SOSD_TP1/Parte2\$'. The user enters './interpretador'. The program outputs a directory listing for '/home/carlos/Desktop' showing directories '..' and 'teste', and files 'novoficheiro.txt', 'forkreturn', and 'ipca.txt'. It then shows the execution of 'apaga /home/carlos/Desktop/file' failing with 'No such file or directory'. Next, 'mostra ../teste' is executed, showing the hex value '0123456789'. Finally, 'termina' is entered, and the program returns to the shell prompt.

```
carlos@carlos:~/Desktop/SOSD/18845_19431_19432_SOSD_TP1/Parte2$ ./interpretador
% lista /home/carlos/Desktop
-----
Diretoria      | ..
Diretoria      | teste
Diretoria      | SOSD
Diretoria      | .
Ficheiro       | novoficheiro.txt
Ficheiro       | forkreturn
Ficheiro       | ipca.txt
-----
Terminou o comando lista com codigo 0
% apaga /home/carlos/Desktop/file
Erro ao apagar o ficheiro: No such file or directory
Terminou o comando apaga com codigo 1
% mostra ../teste
0123456789

Terminou o comando mostra com codigo 0
% termina
carlos@carlos:~/Desktop/SOSD/18845_19431_19432_SOSD_TP1/Parte2$
```

Figura 2: Funcionamento do interpretador

```

int main(int argc, char *argv[])
{
    system("clear");
    char input[600];
    char cmd[12] = "";
    char a1[100] = "", a2[100] = "";
    int count = -1;
    int result;

    while (1)
    {
        printf("%s ");
        fgets(input, 600, stdin);
        count = sscanf(input, "%s %s %s", cmd, a1, a2);

        if (count < 1)
        {
            continue;
        }
        else if (count == 3) {
            char *args[] = { cmd, a1, a2, NULL };
            result = mysystem(cmd, args);
        }
        else{
            char *args[] = { cmd, a1, NULL };
            result = mysystem(cmd, args);
        }
        strcpy(a1, "");
        strcpy(a2, "");
        printf("\033[01;33m");
        printf("Terminou o comando %s com codigo %d\n", cmd, result);
        printf("\033[0m");
    }
    return 0;
}

```

Figura 3: Código Main do Interpretador

Na função main dentro de um while é esperado o input do utilizador para executar a função mysystem.

```

int mysystem(char* cmd, char* args[]) {
    int result;
    int f=0;
    char *path;

    if (strcmp(cmd, "termina") == 0)
    {
        exit(EXIT_SUCCESS);
    }

    strcpy( path, "../Parte1/" );
    strcat( path, cmd );

    f=fork();
    if (f==-1) {
        perror("Erro na criação de um novo processo");
        return EXIT_FAILURE;
    }

    if (f == 0) {
        execv(path, args);
        perror("Erro ao executar o comando");
        exit(EXIT_FAILURE);
    }

    wait(&result);
    return WEXITSTATUS(result);
}

```

Figura 4: Código da Função mysystem

Na função mysystem caso o input do utilizador não seja "termina" que encerra o programa, a função cria um novo processo filho que trata de executar o comando solicitado através da função execv. Entretanto, o processo-pai mantém-se à espera que o processo-filho acabe e devolva o código de terminação com que finalizou para poder informar o utilizador se ocorreu algum erro.

Parte 3

Análise de cópia de ficheiros entre máquinas virtuais

- a) Configure a sua máquina virtual, de modo a que consiga comunicar com o host físico (máquina real).

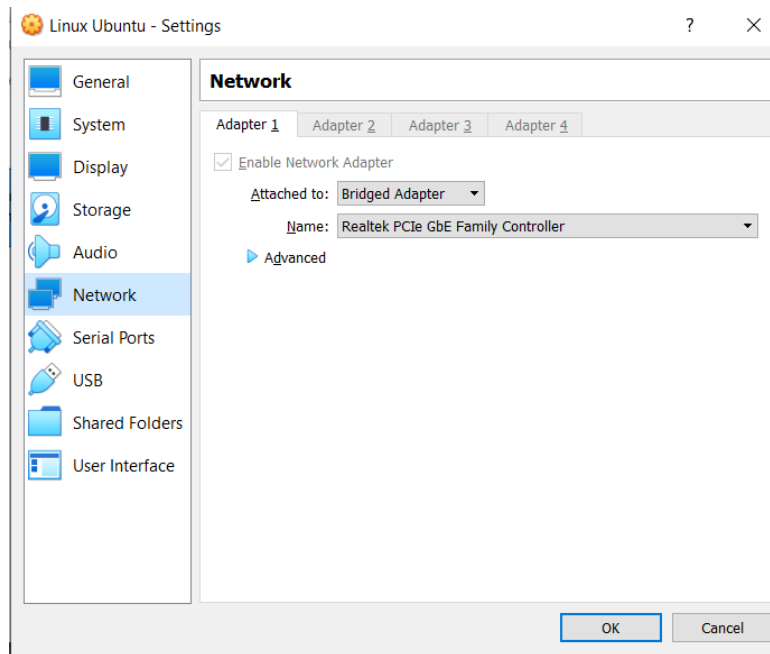


Figura 5: Configuração da máquina Virtual para "BRIDGED ADAPTER"

Para que a máquina virtual conseguisse comunicar com o host foi necessário configurar a rede da máquina virtual para "Bridged Adapter". Assim, foi possível realizar o comando ping da máquina virtual para o host e do host para a máquina virtual com sucesso. Com o comando "ip address show" na máquina virtual e "ipconfig" no host foi possível confirmar o ip de ambas as máquinas.


```
carlos@carlos-VirtualBox: ~/Desktop
carlos@carlos-VirtualBox:~/Desktop$ ping 192.168.1.73
PING 192.168.1.73 (192.168.1.73) 56(84) bytes of data:
64 bytes from 192.168.1.73: icmp_seq=1 ttl=128 time=0.273 ms
64 bytes from 192.168.1.73: icmp_seq=2 ttl=128 time=0.257 ms
64 bytes from 192.168.1.73: icmp_seq=3 ttl=128 time=0.247 ms
64 bytes from 192.168.1.73: icmp_seq=4 ttl=128 time=0.228 ms
^C
--- 192.168.1.73 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3054ms
rtt min/avg/max/mdev = 0.228/0.251/0.273/0.016 ms
carlos@carlos-VirtualBox:~/Desktop$ ip address show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:c3:b3:c0 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.87/24 brd 192.168.1.255 scope global dynamic noprefixroute enp0s3
        valid_lft 3007sec preferred_lft 3007sec
    inet6 2001:818:dad0:8800:7989:f1bc:b6ea:bab4/64 scope global temporary dynamic
        valid_lft 86371sec preferred_lft 3571sec
    inet6 2001:818:dad0:8800:9cd1:3d73:74fa:64e2/64 scope global dynamic mngtppaddr noprefixroute
        valid_lft 86371sec preferred_lft 3571sec
```

Figura 6: Execução do comando ping - *LINUX*

```
C:\WINDOWS\system32>ping 192.168.1.87

Pinging 192.168.1.87 with 32 bytes of data:
Reply from 192.168.1.87: bytes=32 time<1ms TTL=64
Reply from 192.168.1.87: bytes=32 time<1ms TTL=64
Reply from 192.168.1.87: bytes=32 time<1ms TTL=64
Reply from 192.168.1.87: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.1.87:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\WINDOWS\system32>ipconfig

Windows IP Configuration

Ethernet adapter Ethernet 2:

    Connection-specific DNS Suffix  . : 
    Link-local IPv6 Address . . . . . : fe80::650a:bf6b:17b8:c125%11
    Autoconfiguration IPv4 Address. . : 169.254.193.37
    Subnet Mask . . . . . : 255.255.0.0
    Default Gateway . . . . . : 

Ethernet adapter Ethernet:

    Connection-specific DNS Suffix  . : lan
    IPv6 Address. . . . . : 2001:818:dad0:8800:f065:be96:51c4:23f0
    Temporary IPv6 Address. . . . . : 2001:818:dad0:8800:3c8c:9669:8e51:a357
    Link-local IPv6 Address . . . . . : fe80::f065:be96:51c4:23f0%5
    IPv4 Address. . . . . : 192.168.1.73
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : fe80::1%5
                                192.168.1.1
```

Figura 7: Execução do comando ping - *WINDOWS*

b) Recorrendo ao comando iperf3 mostre as diferenças de transferências entre a máquina real e virtual, usando tcp e udp.

```

carlos@carlos:~$ iperf3 -c 192.168.1.64
Connecting to host 192.168.1.64, port 5201
[ 5] local 192.168.1.73 port 47420 connected to 192.168.1.64 port 5201
[ ID] Interval      Transfer    Bitrate      Retr  Cwnd
[ 5] 0.00-1.00 sec  11.7 MBytes 97.7 Mbits/sec  0    64.2 KBytes
[ 5] 1.00-2.00 sec  11.2 MBytes 94.3 Mbits/sec  0    68.4 KBytes
[ 5] 2.00-3.00 sec  11.3 MBytes 95.1 Mbits/sec  0    68.4 KBytes
[ 5] 3.00-4.00 sec  11.2 MBytes 94.0 Mbits/sec  0    68.4 KBytes
[ 5] 4.00-5.00 sec  11.3 MBytes 95.1 Mbits/sec  0    68.4 KBytes
[ 5] 5.00-6.00 sec  11.2 MBytes 94.0 Mbits/sec  0    68.4 KBytes
[ 5] 6.00-7.00 sec  11.3 MBytes 95.1 Mbits/sec  0    68.4 KBytes
[ 5] 7.00-8.00 sec  11.3 MBytes 95.1 Mbits/sec  0    68.4 KBytes
[ 5] 8.00-9.00 sec  11.2 MBytes 93.8 Mbits/sec  0    68.4 KBytes
[ 5] 9.00-10.00 sec 11.3 MBytes 95.1 Mbits/sec  0    68.4 KBytes
-----
[ ID] Interval      Transfer    Bitrate      Retr
[ 5] 0.00-10.00 sec 113 MBytes 94.9 Mbits/sec  0
[ 5] 0.00-10.00 sec 113 MBytes 94.7 Mbits/sec
iperf Done.
carlos@carlos:~$

```

Figura 8: Execução do comando iperf - *CLIENTE*

Recorreu-se ao comando iperf3 para mostrar as diferenças de transferência entre a máquina virtual e a real Após 10 segundos as informações que são mostradas, na imagem acima. Neste exemplo, em 10 segundos foram transferidos 113 MBytes, atingindo a velocidade de média de 94,9 Mb/s. Conclui-se também que a taxa de envio foi superior a taxa de recepção.

```

Administrator: Linha de comandos - D:\Transferências\iperf-3.1.2-win64\iperf3.exe -s
C:\Windows\system32>D:\Transferências\iperf-3.1.2-win64\iperf3.exe -s
Server listening on 5201
Accepted connection from 192.168.1.64 port 5201 connected to 192.168.1.73 port 47420
[ 5] local 192.168.1.64 port 5201 connected to 192.168.1.73 port 47420
[ ID] Interval      Transfer    Bandwidth
[ 5] 0.00-1.00 sec  11.3 MBytes 94.7 Mbits/sec
[ 5] 1.00-2.00 sec  11.3 MBytes 94.6 Mbits/sec
[ 5] 2.00-3.00 sec  11.3 MBytes 94.4 Mbits/sec
[ 5] 3.00-4.00 sec  11.3 MBytes 94.8 Mbits/sec
[ 5] 4.00-5.00 sec  11.2 MBytes 94.2 Mbits/sec
[ 5] 5.00-6.00 sec  11.3 MBytes 94.7 Mbits/sec
[ 5] 6.00-7.00 sec  11.3 MBytes 94.7 Mbits/sec
[ 5] 7.00-8.00 sec  11.3 MBytes 94.9 Mbits/sec
[ 5] 8.00-9.00 sec  11.3 MBytes 94.9 Mbits/sec
[ 5] 9.00-10.00 sec 11.3 MBytes 94.6 Mbits/sec
[ 5] 10.00-10.00 sec 32.8 KBytes 96.5 Mbits/sec
-----
[ ID] Interval      Transfer    Bandwidth
[ 5] 0.00-10.00 sec  0.00 Bytes  0.00 bits/sec
[ 5] 0.00-10.00 sec 113 MBytes 94.7 Mbits/sec
Server listening on 5201

```

Figura 9: Execução do comando iperf - *SERVIDOR*

¹

¹iperf3 -c , iperf3 -s - Inicia o Iperf como Cliente e Servidor, respetivamente

c) Instale um servidor web (nginx ou apache) e mostre os resultados de um teste de carga ao “url /”, simulando 10 clientes em simultâneo e 500 pedidos cada um, utilizando o utilitário ab ou jmeter, sendo que o jmeter terá um acréscimo de 0.3 valores.

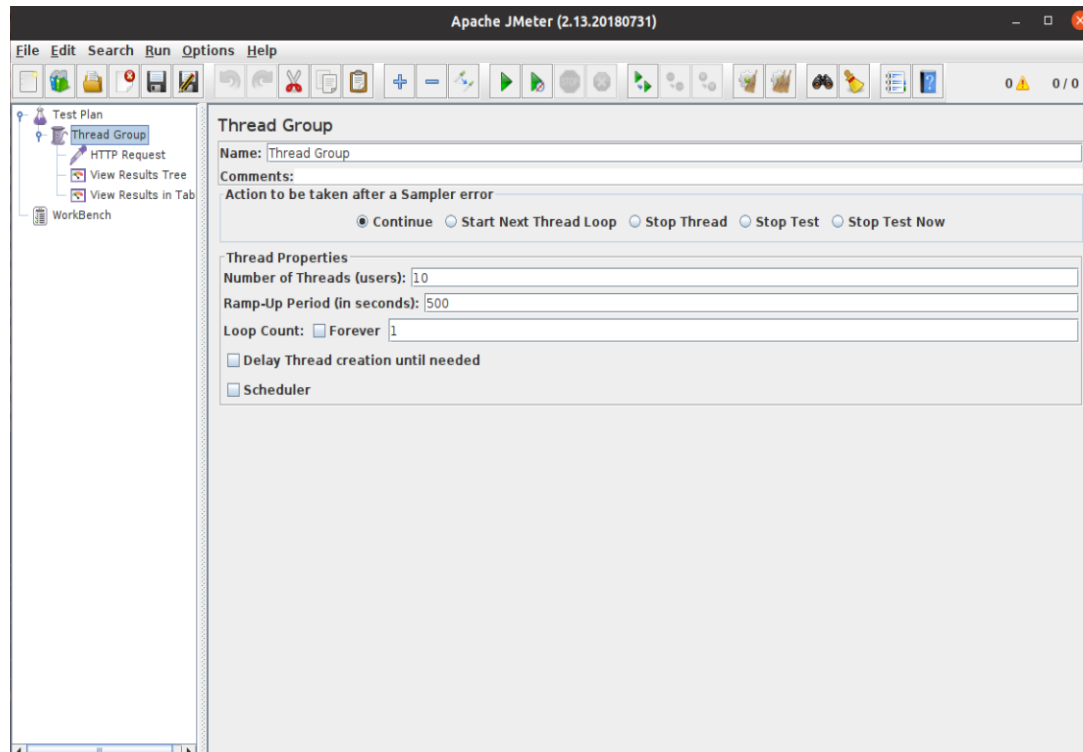


Figura 10: Servidor Web Apache e Utilitário JMeter

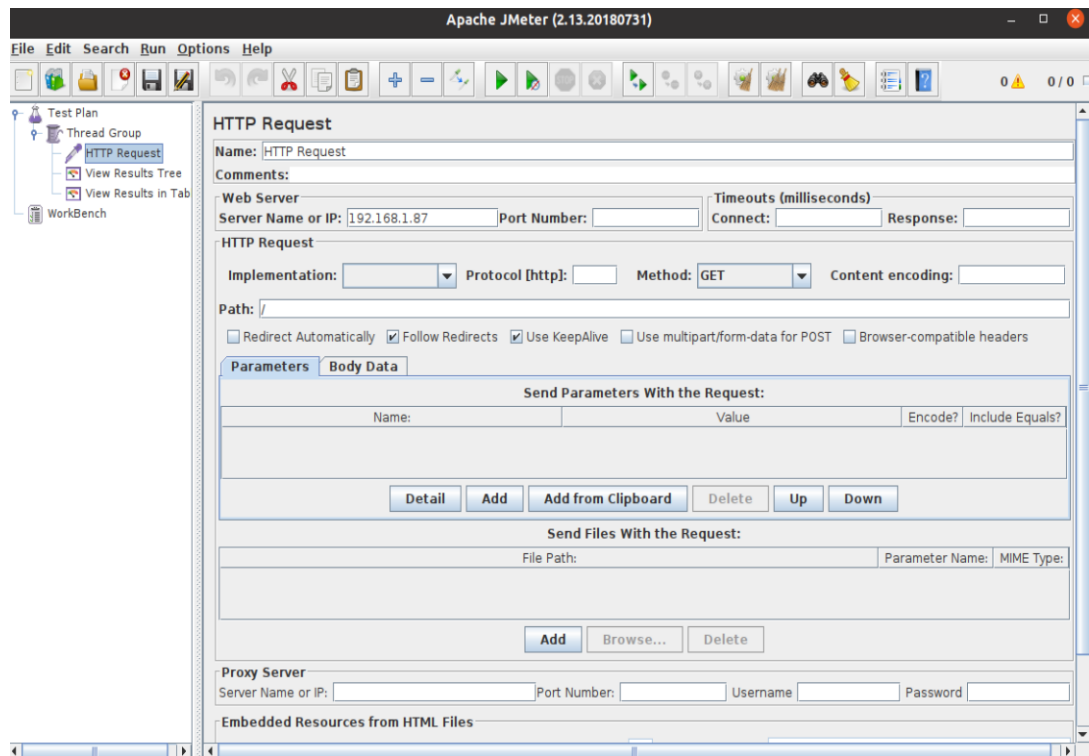


Figura 11: Configurações efetuadas ao JMeter

Test Plan.jmx (/home/carlos/Desktop/Test Plan.jmx) - Apache JMeter (2.13.20180731)

File Edit Search Run Options Help

Test Plan
Thread Group
HTTP Request
View Results in Table
View Results Tree
WorkBench

View Results in Table

Name: View Results in Table

Comments:

Write results to file / Read from file

Filename: Browse...

Log/Display Only: ☐ Errors ☐ Successes

Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Latency	Connect Tim...
4973	16:35:48.173	Thread Group 1-10	HTTP Request	1		11228	1	0
4974	16:35:48.192	Thread Group 1-10	HTTP Request	1		11228	1	0
4975	16:35:48.201	Thread Group 1-10	HTTP Request	7		11228	7	0
4976	16:35:48.212	Thread Group 1-10	HTTP Request	0		11228	0	0
4977	16:35:48.216	Thread Group 1-10	HTTP Request	4		11228	4	0
4978	16:35:48.226	Thread Group 1-10	HTTP Request	0		11228	0	0
4979	16:35:48.248	Thread Group 1-10	HTTP Request	0		11228	0	0
4980	16:35:48.250	Thread Group 1-10	HTTP Request	2		11228	2	0
4981	16:35:48.283	Thread Group 1-10	HTTP Request	0		11228	0	0
4982	16:35:48.290	Thread Group 1-10	HTTP Request	0		11228	0	0
4983	16:35:48.292	Thread Group 1-10	HTTP Request	4		11228	4	0
4984	16:35:48.298	Thread Group 1-10	HTTP Request	0		11228	0	0
4985	16:35:48.304	Thread Group 1-10	HTTP Request	0		11228	0	0
4986	16:35:48.306	Thread Group 1-10	HTTP Request	1		11228	1	0
4987	16:35:48.316	Thread Group 1-10	HTTP Request	1		11228	1	0
4988	16:35:48.335	Thread Group 1-10	HTTP Request	0		11228	0	0
4989	16:35:48.351	Thread Group 1-10	HTTP Request	1		11228	1	0
4990	16:35:48.366	Thread Group 1-10	HTTP Request	2		11228	2	0
4991	16:35:48.388	Thread Group 1-10	HTTP Request	1		11228	1	0
4992	16:35:48.398	Thread Group 1-10	HTTP Request	0		11228	0	0
4993	16:35:48.408	Thread Group 1-10	HTTP Request	1		11228	1	0
4994	16:35:48.428	Thread Group 1-10	HTTP Request	1		11228	1	0
4995	16:35:48.438	Thread Group 1-10	HTTP Request	3		11228	3	0
4996	16:35:48.452	Thread Group 1-10	HTTP Request	1		11227	1	0
4997	16:35:48.466	Thread Group 1-10	HTTP Request	1		11227	1	0
4998	16:35:48.479	Thread Group 1-10	HTTP Request	1		11227	1	0
4999	16:35:48.497	Thread Group 1-10	HTTP Request	1		11227	1	0
5000	16:35:48.509	Thread Group 1-10	HTTP Request	1		11227	1	0

☒ Scroll automatically? ☐ Child samples? No of Samples 5000 Latest Sample 1 Average 0 Deviation 1

Figura 12: Resultados apresentados em Tabela

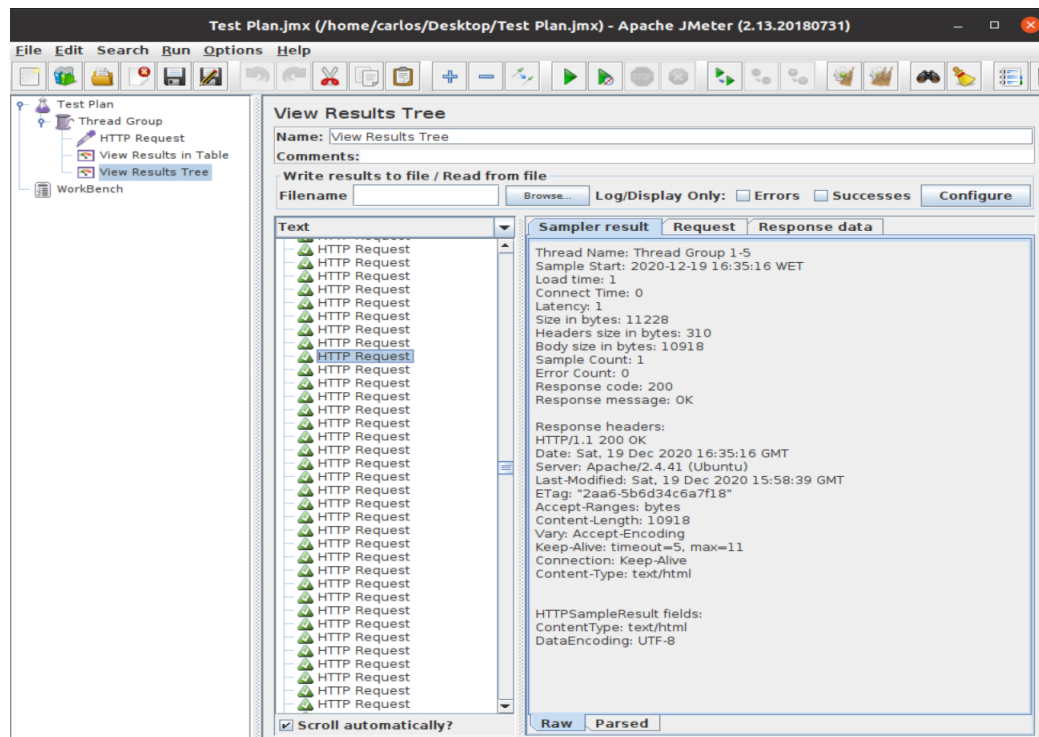


Figura 13: Resultados apresentados em Árvore

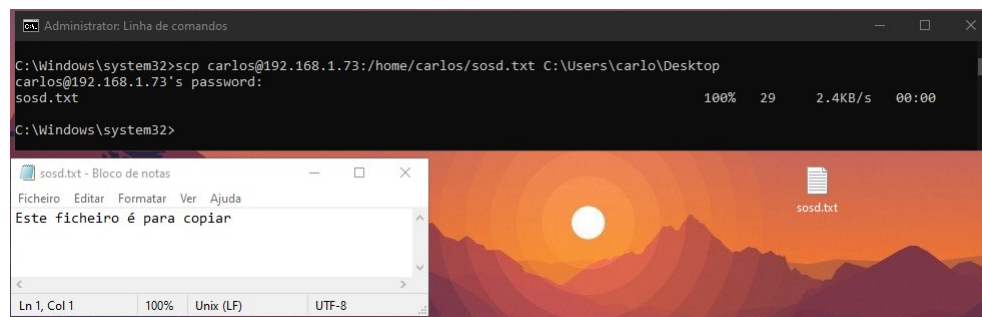
Depois de instalado o servidor web apache foi realizado um teste de carga através do programa apache jmeter. Primeiramente foi configurado o "thread group" com 10 clientes e cada um com 500 pedidos. De seguido foi indicado qual o ip do servidor ao qual estava a ser feito o teste e o seu url. No fim foi possível verificar os resultados em tabela ou em árvore.

Em tabela, é possível ver a lista de resultados e analisar o tempo, os bytes e a latência de cada "request".

Em árvore podemos ver os resultados de forma mais detalhada. Todos os requests tiveram uma latência mínima e um número de bytes transferidos constante.

d) Tendo como base o problema de copiar um ficheiro da primeira máquina virtual para a segunda máquina virtual (este ficheiro deverá ser criado na diretoria /home/<nomeutilizador>/, com o nome sosd.txt e com o conteúdo “este ficheiro é para copiar”), indique os comandos que permitem realizar esta operação. Faça uso das seguintes sugestões de comandos para apresentar até 3 possíveis soluções distintas: 1) scp 2) dd, nc, pipe (|) 3) cat, ssh, pipe (|) Apresente o comando para cada uma das três possíveis soluções com uma descrição de cada uma delas, indicando qual a melhor em termos de utilização de recursos e rapidez. vspace2 em

```
carlos@carlos:~$ pwd
/home/carlos
carlos@carlos:~$ touch sosd.txt
carlos@carlos:~$ gedit sosd.txt
carlos@carlos:~$ cat sosd.txt
Este ficheiro é para copiar
carlos@carlos:~$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eno1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether e4:e7:49:50:a3:b7 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.73/24 brd 192.168.1.255 scope global dynamic noprefixroute eno1
        valid_lft 2860sec preferred_lft 2860sec
    inet6 2001:818:dad0:8800:4d00:6688:35e0:a417/64 scope global temporary dynamic
        valid_lft 86371sec preferred_lft 3571sec
    inet6 2001:818:dad0:8800:5271:c337:ea5b:1bc/64 scope global dynamic mngtmpaddr noprefixroute
        valid_lft 86371sec preferred_lft 3571sec
    inet6 fe80::78fd:a841:a0f7:4054/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
3: wlo1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 0c:96:e6:73:cf:59 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.69/24 brd 192.168.1.255 scope global dynamic noprefixroute wlo1
        valid_lft 2860sec preferred_lft 2860sec
    inet6 2001:818:dad0:8800:211c:5ea4:33bd:226e/64 scope global temporary dynamic
        valid_lft 86371sec preferred_lft 3571sec
    inet6 2001:818:dad0:8800:a142:66f:55b:c506/64 scope global dynamic mngtmpaddr noprefixroute
        valid_lft 86371sec preferred_lft 3571sec
    inet6 fe80::b53c:287c:78ea:41e3/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
carlos@carlos:~$
```



Com o objetivo de copiar ficheiros entre máquinas foi necessário instalar o scp-server na máquina origem, na máquina destino, através do scp-client e do comando (scp [utilizador]@[ip_origem]:[caminho_ficheiro] [destino_ficheiro]), foi copiado o ficheiro sendo assim possível obter uma cópia do ficheiro na máquina destino.

Conclusão

Na nossa opinião foi muito interessante o desenvolvimento deste projeto, pois potencializou a experiência do desenvolvimento de Software. Assimilar os conteúdos da Unidade Curricular, desenvolver Capacidades de programação em C

Sentimos que este projeto foi bastante exigente e fez com que nos dedicássemos mais e consequentemente melhorar as nossas capacidades.

Com este Trabalho adquirimos inúmeras valias que nos serão úteis em futuros projetos.

Em suma, abordamos todos os assuntos lecionados e graças a isso conseguimos cumprir os objetivos propostos.

Execução das Tarefas

1. Parte 1
 - a) João Ricardo Nº 18845
 - b) João Ricardo Nº 18845
 - c) Carlos Santos Nº 19432
 - d) João Rodrigues Nº 19431
 - e) João Ricardo Nº 18845
 - f) João Rodrigues Nº 19431
2. Parte 2
 - Carlos Santos Nº 19432
 - João Ricardo Nº 18845
3. Parte 3
 - Carlos Santos Nº 19432