UNIVERSIDADE UNIAVAN
SISTEMAS DA INFORMAÇÃO

EDUARDO ZORNITTA DA SILVA

PROCESSOS, MEMÓRIA E GERENCIAMENTO DE ARQUIVOS

BALNEÁRIO CAMBORIÚ
2025

Bom dia, hoje iremos fazer alguns experimento com o sistema Linux (Ubuntu) em 4 passos, muito simples, dividido em sub etapas respectivas para que possamos extrair o melhor desses experimentos, vamos começar:

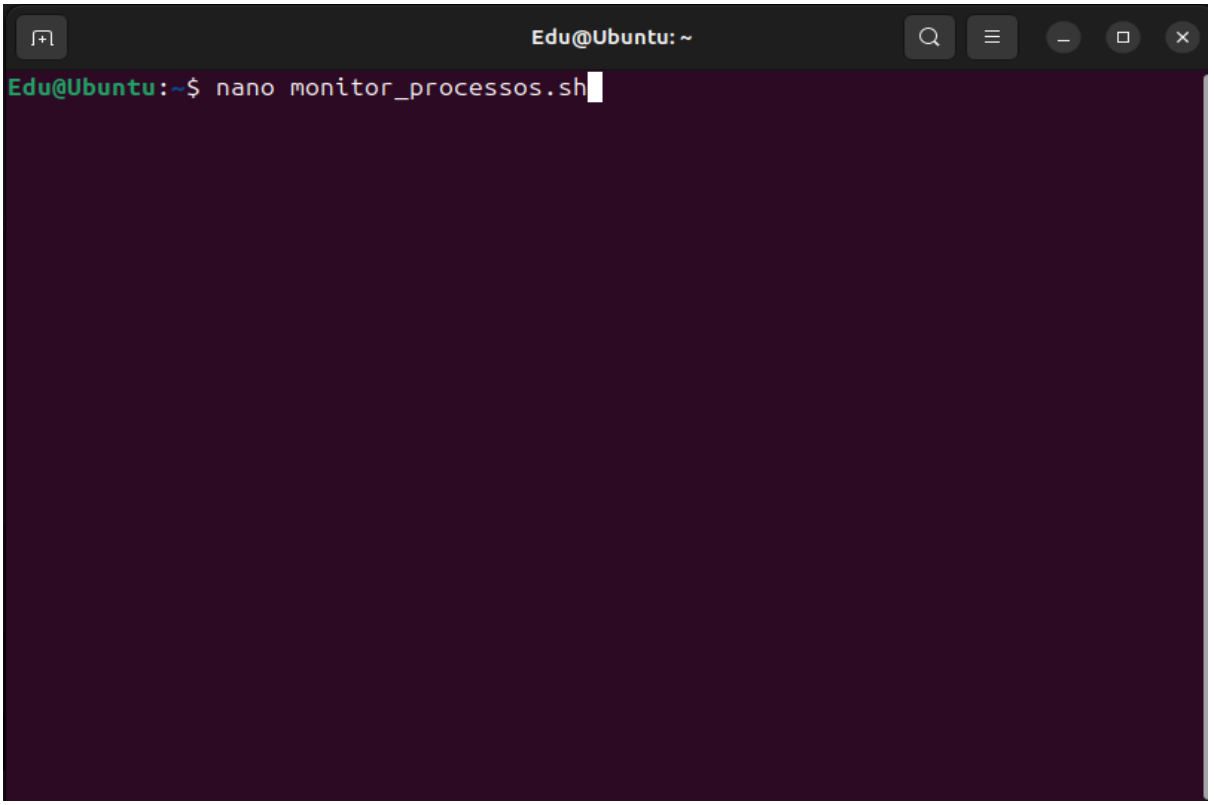
01. Monitorando Processos e Recursos do Sistema

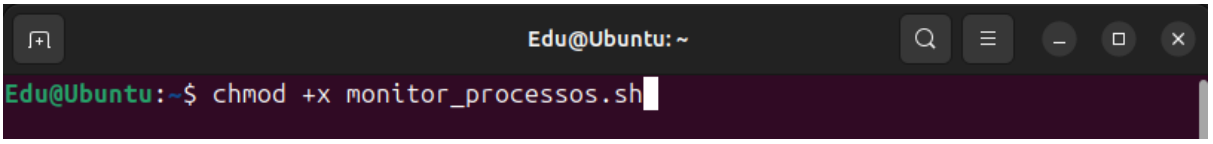
1. Explorar gerenciamento de processos, CPU, memória e I/O:

- 1.1 a) Escrever um pequeno script (Bash) que liste os 5 processos que mais consomem CPU e memória. (Segue o código abaixo):

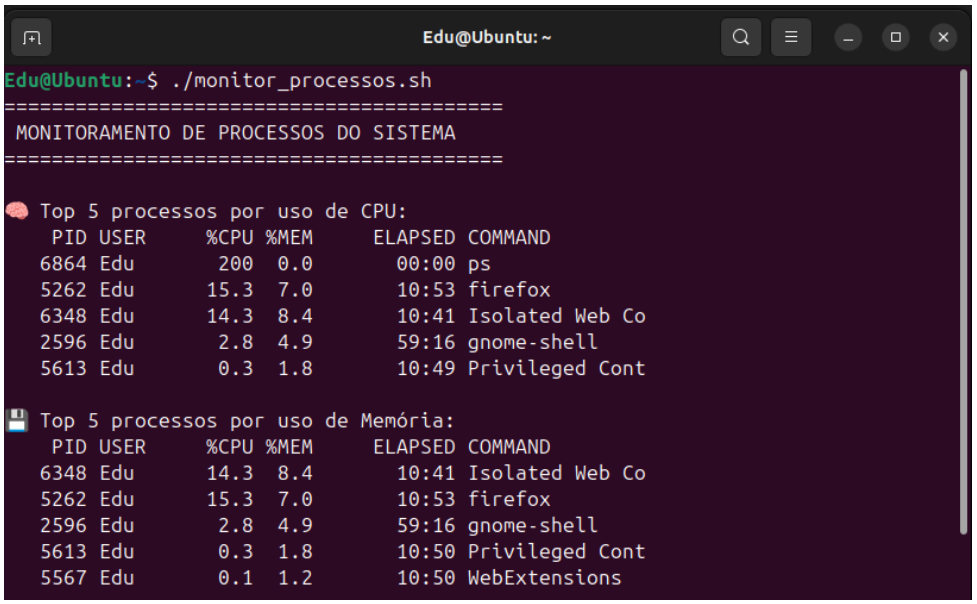
```
# ======= CONFIGURAÇÕES =======
LIMITE_CPU=50.0   # Alerta se algum processo usar mais que 50% da CPU
LIMITE_MEM=30.0   # Alerta se algum processo usar mais que 30% da Memória

echo "========================================="
echo " MONITORAMENTO DE PROCESSOS DO SISTEMA"
echo "========================================="
echo
echo "🧠 Top 5 processos por uso de CPU:"
ps -eo pid,user,pcpu,pmem,etime,comm --sort=-pcpu | head -n 6
echo
echo "💾 Top 5 processos por uso de Memória:"
ps -eo pid,user,pcpu,pmem,etime,comm --sort=-pmem | head -n 6
echo
echo "========================================="
echo "⏱️  Verificação de limites"
echo "========================================="
# Checar se há algum processo acima dos limites definidos
ALTO_CPU=$(ps -eo pid,user,pcpu,comm --sort=-pcpu | awk -v limite="$LIMITE_CPU" '$3>limite {print}')
ALTO_MEM=$(ps -eo pid,user,pmem,comm --sort=-pmem | awk -v limite="$LIMITE_MEM" '$3>limite {print}')

if [ -n "$ALTO_CPU" ]; then
    echo "⚠️ ALERTA: Processos com uso de CPU acima de $LIMITE_CPU%"
    echo "$ALTO_CPU"
    echo
fi

if [ -n "$ALTO_MEM" ]; then
    echo "⚠️ ALERTA: Processos com uso de Memória acima de $LIMITE_MEM%"
    echo "$ALTO_MEM"
    echo
fi
echo "✅ Monitoramento concluído!"
echo "========================================="
```

Agora é a hora do teste, começando com a criação do arquivo .bash, inicializando e atribuindo a permissão de execução com o chmod +x monitor_processos.sh no código no nano:





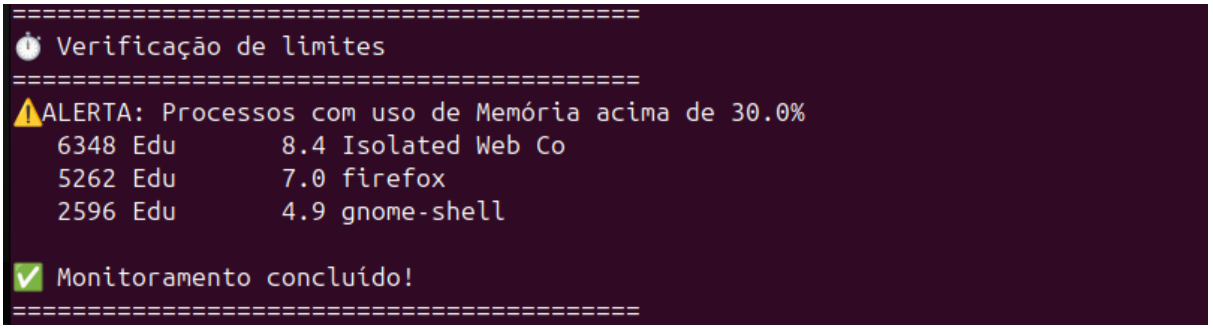
1.2 b) Exibir informações como PID, usuário, %CPU, %MEM, tempo de execução:



Verifiquemos que os processos que mais consomem da cpu e da ram, respectivamente, são:

ps e firefox - Na CPU
Isolated Web Co e firefox - No uso de memória RAM

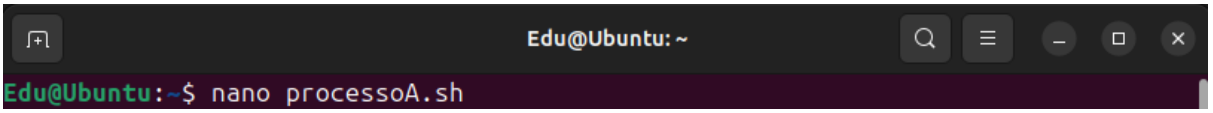
1.3 Agora o campo opcional o alerta se algum processo ultrapassar determinado limite de CPU ou memória, dado pelo campo a seguir:

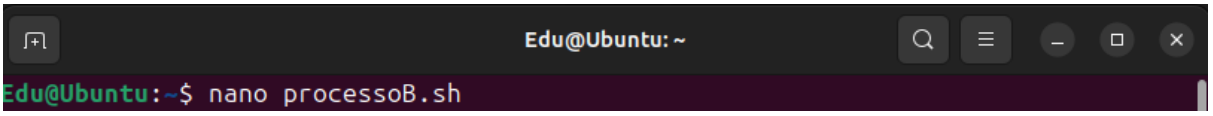


Com isso concluímos a primeira etapa, vamos seguindo.

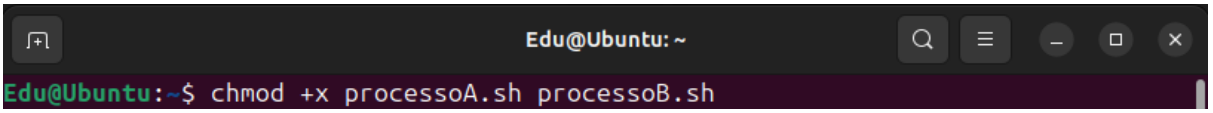
02. Comunicação entre Processos (IPC)
Entender como os processos podem se comunicar

Primeiro de tudo precisamos criar os arquivos nano processoA.sh e nano processoB.sh:

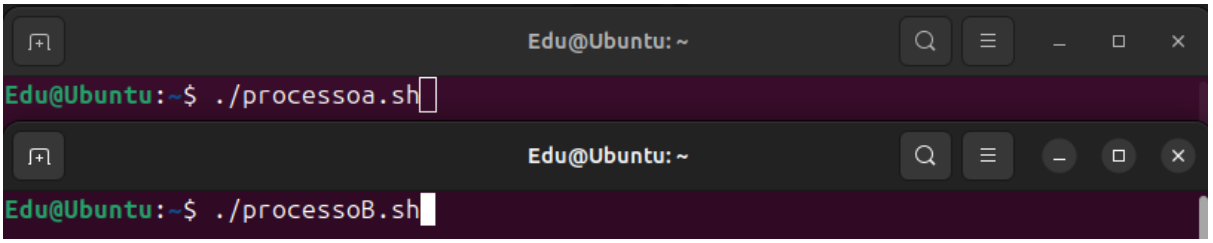




Precisamos liberar as permissões do processos

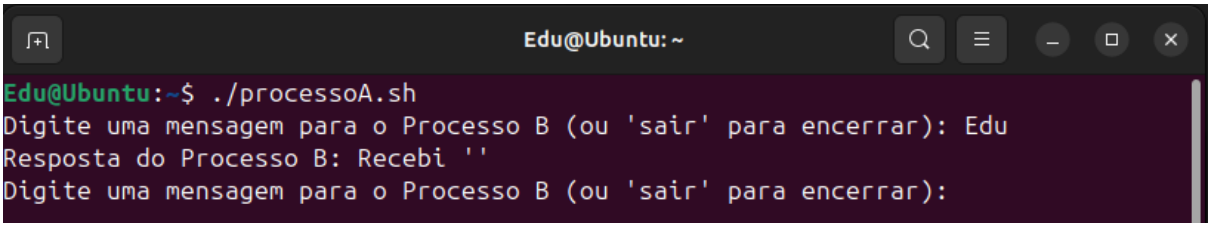


Abrir os processos em terminais diferentes

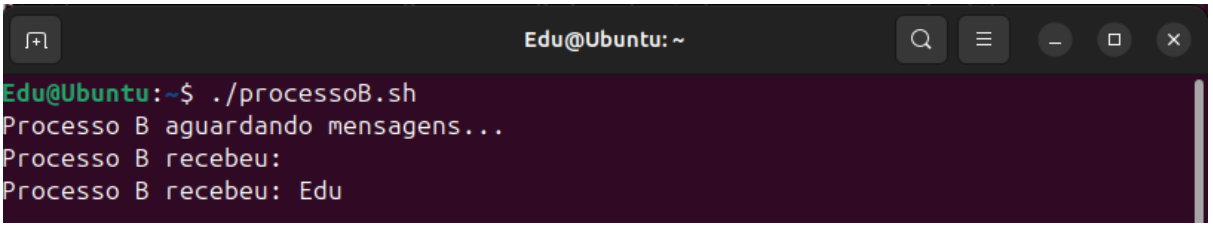


Iniciamos, primeiramente, o processoB.sh para que espere as mensagens e o processoA.sh, logo após, para que enviemos a mensagem

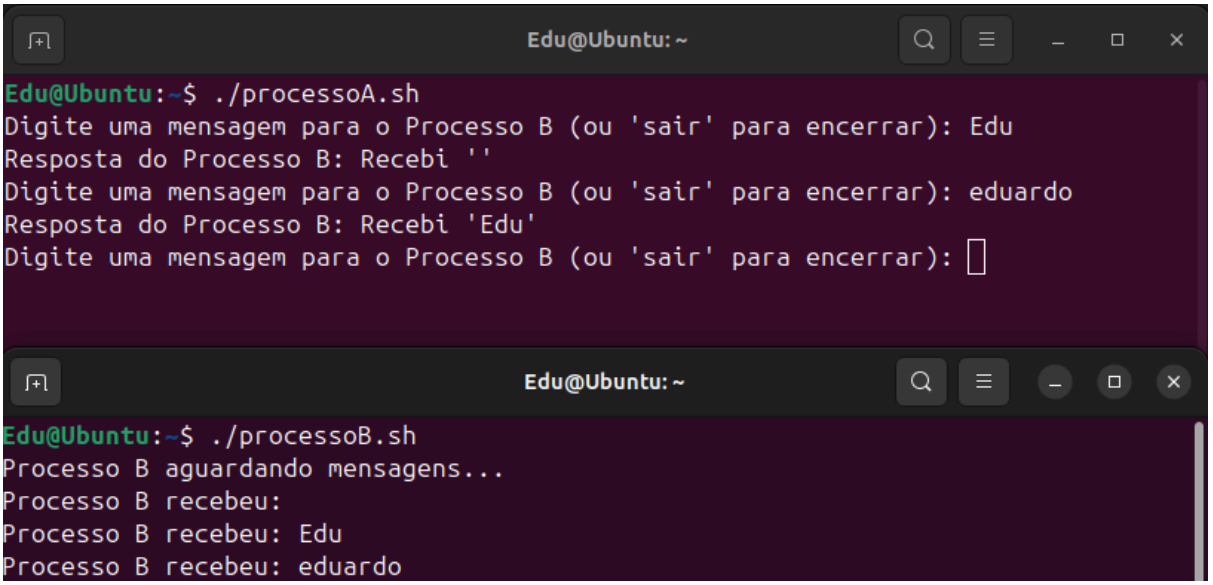
Agora é apenas executar os códigos, onde o processoA.sh envia a mensagem escrita.



E o processoB.sh recebe e apresenta a mensagem enviada.



No processo, é retornado a rotina das mensagens recebidas no conteúdo anterior da mensagem já recebida no processoA.sh, como na imagem a seguir, com isso o processo retorna o que o processoB.sh já havia recebido posteriormente enquanto aguarda o conteúdo da próxima mensagem e na tela do processo B, ele já mostra o que recebeu:



03. Exercício de Memória
Mostrar alocação de memória e consumo de processos

3.1 - Utilizando o código em C:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main() {
    const size_t QUATRO_GB = 4ULL * 1024 * 1024 * 1024;

    printf("Alocando 4 GB de RAM...\n");

    char *grande_buffer = (char *)malloc(QUATRO_GB);

    if (grande_buffer == NULL) {
        printf("ERRO: Nao foi possivel alocar 4 GB!\n");
        return 1;
    }

    printf("Preenchendo toda a memoria para forcar alocacao fisica...\n");

    for (size_t i = 0; i < QUATRO_GB; i += 4096) {
        grande_buffer[i] = (char)(i % 256);

        if (i % (512ULL * 1024 * 1024) == 0 && i != 0) {
            printf("Preenchidos: %.2f GB\n", (double)i / (1024 * 1024 * 1024));
            fflush(stdout);
        }
    }

    printf("\nCOMPLETO: 4 GB realmente alocados na RAM fisica!\n");
    printf("Verifique o consumo no top/htop...\n");
    printf("Pressione ENTER para liberar a memoria...");
    getchar();

    free(grande_buffer);
    printf("Memoria liberada!\n");

    return 0;
}
```

```
Edu@Ubuntu: ~
Edu@Ubuntu:~$ nano memoria.c
```

Compilamos:

```
Edu@Ubuntu: ~
Edu@Ubuntu:~$ gcc memoria.c -o memoria
```

E Executamos:

```
Edu@Ubuntu: ~
Edu@Ubuntu:~$ nano memoria.c
Edu@Ubuntu:~$ gcc memoria.c -o memoria
Edu@Ubuntu:~$ ./memoria
Alocando 4 GB de RAM...
Preenchendo toda a memoria para forcar alocacao fisica...
Preenchidos: 0.50 GB
Preenchidos: 1.00 GB
Preenchidos: 1.50 GB
Preenchidos: 2.00 GB
Preenchidos: 2.50 GB
Preenchidos: 3.00 GB
Preenchidos: 3.50 GB

COMPLETO: 4 GB realmente alocados na RAM fisica!
Verifique o consumo no top/htop...
Pressione ENTER para liberar a memoria...
```

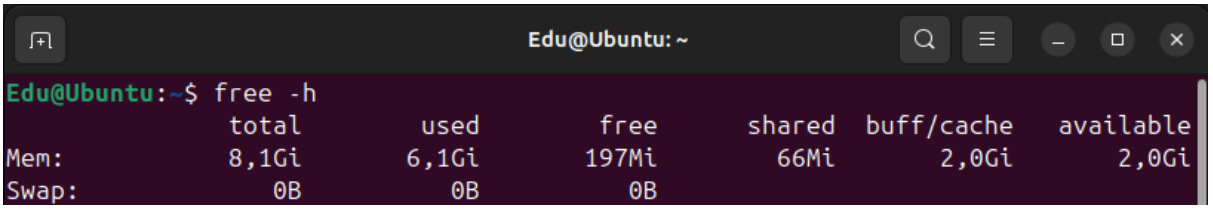
Aplicamos as seguintes etapas dos processos:

3.2 - Monitorar consumo de memória usando top (em outro terminal)

```
Edu@Ubuntu: ~
top - 12:39:31 up  3:34,  2 users,  load average: 0,35, 0,22, 0,13
Tasks: 274 total,   1 running, 273 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1,1 us,  1,2 sy,  0,0 ni, 96,6 id,  0,0 wa,  0,0 hi,  1,0 si,  0,0 st
MiB Mem :   8344,6 total,    356,4 free,   6313,7 used,   1914,8 buff/cache
MiB Swap:      0,0 total,      0,0 free,      0,0 used.   2030,9 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
2596	Edu	20	0	5248416	444204	164868	S	26,6	5,2	6:08.07	gnome-s+
11776	Edu	20	0	560120	57880	45688	S	1,3	0,7	0:01.13	gnome-t+
5613	Edu	20	0	2513412	155220	97472	S	0,7	1,8	0:13.75	Privile+
6348	Edu	20	0	3333600	708448	113352	S	0,7	8,3	3:24.61	Isolate+
11185	root	20	0	0	0	0	I	0,7	0,0	0:00.26	kworker+
18	root	20	0	0	0	0	I	0,3	0,0	0:07.38	rcu_pre+
2917	Edu	20	0	462128	11084	7356	S	0,3	0,1	0:07.96	ibus-da+
3494	Edu	20	0	392524	8352	7456	S	0,3	0,1	0:01.64	gvfs-af+
11357	root	20	0	0	0	0	I	0,3	0,0	0:00.15	kworker+
11687	Edu	20	0	2921336	62104	48236	S	0,3	0,7	0:00.56	gjs
1	root	20	0	25828	15632	10512	S	0,0	0,2	0:05.31	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.08	kthreadd
3	root	20	0	0	0	0	S	0,0	0,0	0:00.00	pool_wo+
4	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker+
5	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker+
6	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker+
7	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker+

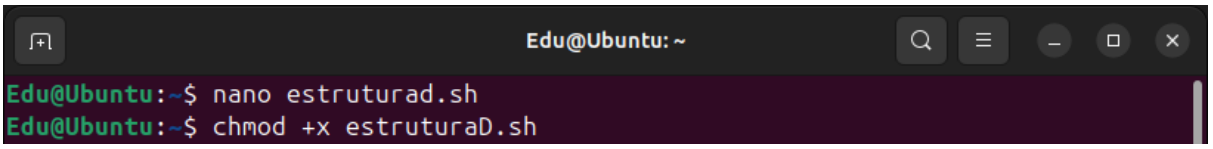
3.3 Analisar comportamento: quando o sistema começa a usar swap, se há impacto na performance utilizando o comando free -h.



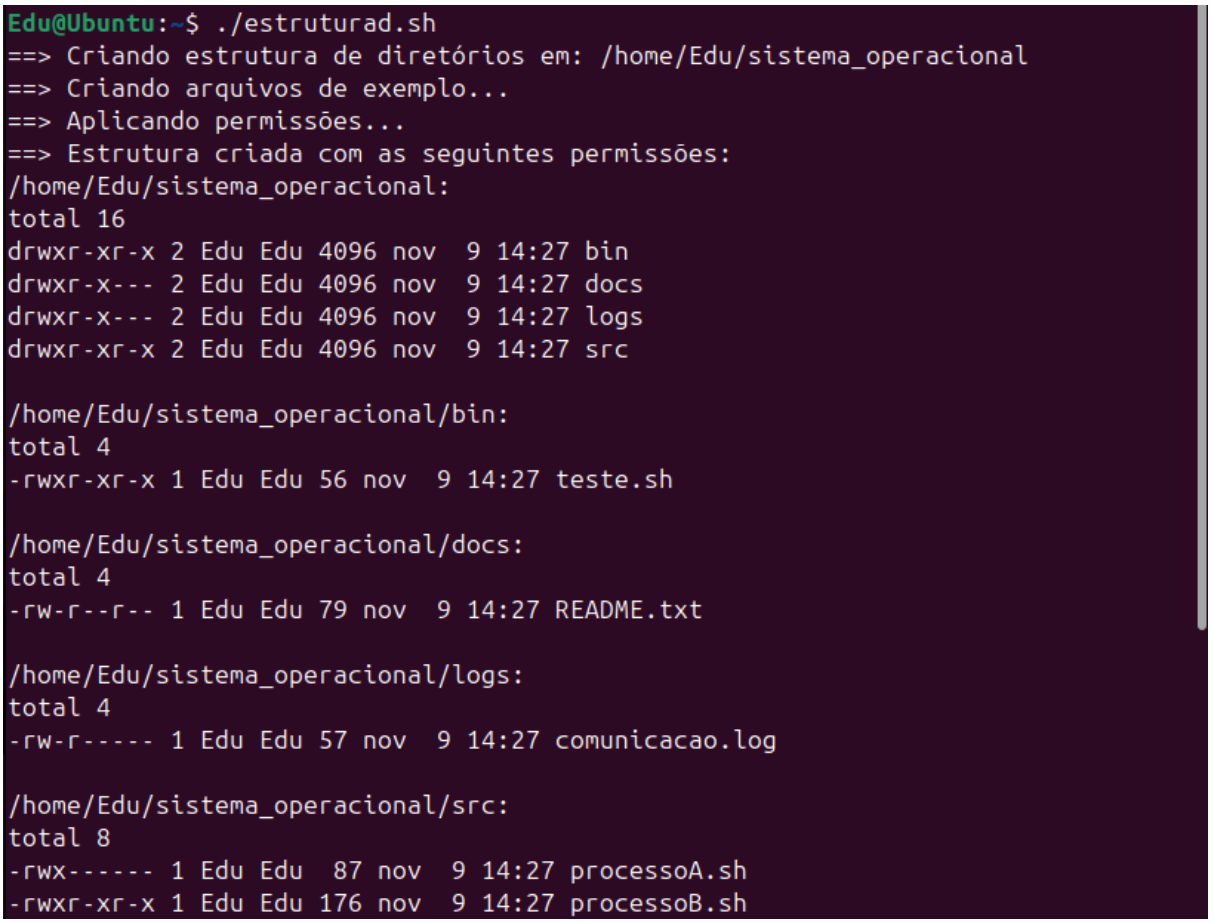
Com isso, vamos para o último plano de verificação do sistema o:

04. Gerenciamento de Arquivos:
Trabalhar com estrutura de diretórios, permissões e operações de arquivos.

4.1 Começamos com o de sempre, criando o arquivo estrutura.sh e atribuindo as permissões ao mesmo, com:



4.2 Após isso temos a execução do mesmo:



```
==> Arquivos modificados nas últimas 24 horas:
-rw-r--r-- 1 Edu Edu 79 nov  9 14:27 /home/Edu/sistema_operacional/docs/README.txt
-rw-r----- 1 Edu Edu 57 nov  9 14:27 /home/Edu/sistema_operacional/logs/comunicacao.log
-rwxr-xr-x 1 Edu Edu 176 nov  9 14:27 /home/Edu/sistema_operacional/src/processoB.sh
-rwx------ 1 Edu Edu 87 nov  9 14:27 /home/Edu/sistema_operacional/src/processoA.sh
-rwxr-xr-x 1 Edu Edu 56 nov  9 14:27 /home/Edu/sistema_operacional/bin/teste.sh
==> Script finalizado com sucesso!
```

Utilizando o script:

```
# === 1. Definir caminho base ===
BASE="$HOME/sistema_operacional"

echo "==> Criando estrutura de diretórios em: $BASE"
mkdir -p "$BASE"/{bin,docs,logs,src}

# === 2. Criar arquivos de exemplo ===
echo "==> Criando arquivos de exemplo..."

# Exercício 02 - Comunicação entre processos
echo '#!/bin/bash
echo "Processo A: enviando mensagem para o arquivo..." > /tmp/mensagem.txt' >
"$BASE/src/processoA.sh"

echo '#!/bin/bash
if [ -f /tmp/mensagem.txt ]; then
    echo "Processo B: recebeu a mensagem -> $(cat /tmp/mensagem.txt)"
else
    echo "Processo B: nenhuma mensagem encontrada."
fi' > "$BASE/src/processoB.sh"

# Outros arquivos de exemplo
echo "Documentação do sistema operacional - criado em $(date)" >
"$BASE/docs/README.txt"
echo "Log de execução criado em $(date)" > "$BASE/logs/comunicacao.log"
echo '#!/bin/bash
echo "Executando script de teste em bin..."' > "$BASE/bin/teste.sh"

# === 3. Modificar permissões ===
echo "==> Aplicando permissões..."
chmod 755 "$BASE/bin" "$BASE/src"
chmod 750 "$BASE/docs" "$BASE/logs"

chmod 755 "$BASE/bin/teste.sh"
chmod 700 "$BASE/src/processoA.sh"
chmod 755 "$BASE/src/processoB.sh"
chmod 644 "$BASE/docs/README.txt"
chmod 640 "$BASE/logs/comunicacao.log"
```

```
# === 4. Verificar estrutura e permissões ===
echo "==> Estrutura criada com as seguintes permissões:"
ls -lR "$BASE"

# === 5. Listar arquivos modificados nas últimas 24 horas ===
echo "==> Arquivos modificados nas últimas 24 horas:"
find "$BASE" -type f -mtime -1 -exec ls -lh {} \;

# === 6. Finalização ===
echo "==> Script finalizado com sucesso!"
```

com isso, podemos ver:

4.2: Crie arquivos dentro desses diretórios com conteúdos de exemplo (usar os exemplos feitos o item 02, ref. a comunicação entre processos)

```
==> Criando estrutura de diretórios em: /home/Edu/sistema_operacional
==> Criando arquivos de exemplo...
```

4.3: Modifique permissões de alguns arquivos (chmod) e verifique com ls -l.

```
==> Estrutura criada com as seguintes permissões:
/home/Edu/sistema_operacional:
total 16
drwxr-xr-x 2 Edu Edu 4096 nov  9 14:27 bin
drwxr-x--- 2 Edu Edu 4096 nov  9 14:27 docs
drwxr-x--- 2 Edu Edu 4096 nov  9 14:27 logs
drwxr-xr-x 2 Edu Edu 4096 nov  9 14:27 src

/home/Edu/sistema_operacional/bin:
total 4
-rwxr-xr-x 1 Edu Edu 56 nov  9 14:27 teste.sh

/home/Edu/sistema_operacional/docs:
total 4
-rw-r--r-- 1 Edu Edu 79 nov  9 14:27 README.txt

/home/Edu/sistema_operacional/logs:
total 4
-rw-r----- 1 Edu Edu 57 nov  9 14:27 comunicacao.log
```

```
/home/Edu/sistema_operacional/src:
total 8
-rwx------ 1 Edu Edu  87 nov  9 14:27 processoA.sh
-rwxr-xr-x 1 Edu Edu 176 nov  9 14:27 processoB.sh
```

4.4: Liste todos os arquivos modificados nas últimas 24 horas.

```
==> Arquivos modificados nas últimas 24 horas:
-rw-r--r-- 1 Edu Edu 79 nov  9 14:27 /home/Edu/sistema_operacional/docs/README.txt
-rw-r----- 1 Edu Edu 57 nov  9 14:27 /home/Edu/sistema_operacional/logs/comunicacao.log
-rwxr-xr-x 1 Edu Edu 176 nov  9 14:27 /home/Edu/sistema_operacional/src/processoB.sh
-rwx------ 1 Edu Edu 87 nov  9 14:27 /home/Edu/sistema_operacional/src/processoA.sh
-rwxr-xr-x 1 Edu Edu 56 nov  9 14:27 /home/Edu/sistema_operacional/bin/teste.sh
==> Script finalizado com sucesso!
```