

Processos

Pedro Cruz

EEL770 – Sistemas Operacionais

Datas importantes



- 02 de Abril
 - Proposta de trabalho
- 09 de Abril
 - Confirmação de proposta
- 07 de Maio
 - Primeira apresentação do trabalho
- 09 de Maio
 - Entrega da primeira lista
- 14 de Maio
 - Primeira prova

- O que vamos falar vale para sistemas de processador/
núcleo único
 - Vale também para cada processador/núcleo de um sistema
com múltiplos processadores/núcleos

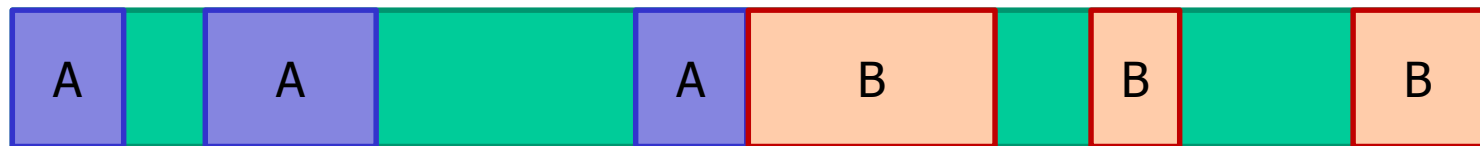
- Abstração para um programa sendo executado
 - Espaço de endereçamento - Contexto
 - Executável
 - Código
 - Dados
 - Informações usadas
 - Pilha
 - Armazenamento local
 - Passagem de parâmetros

- Duas instâncias de um mesmo programa são dois processos diferentes
 - Cada um tem um espaço de endereçamento:
 - Pilha
 - Dados
 - Código*

* Em alguns casos, otimizações são feitas e o código é o mesmo, com contadores de programa diferentes

Subutilização do processador

- Processador é muito mais rápido que outros componentes
 - Processador fica ocioso aguardando os outros componentes
 - Ideia: executar outros processos no tempo ocioso



t →



Pseudoparalelismo



- Diversos processos executados em fatias de tempo
 - Tempo do processador é subdividido e distribuído entre processos
 - Centenas ou dezenas de ms para cada processo
 - (milhões de ciclos de clock)
 - Extrapolação da ideia de reduzir o tempo ocioso
 - Melhorar a experiência do usuário, com a ilusão de paralelismo

- Define as fatias de tempo para cada processo
- Troca os contextos
 - Registradores são carregados com os valores do processo que vai ser executado
 - Contador de programa
 - Apontador de pilha
 - *Flags* e ambiente

Gerenciamento de processos



- Criação de processos
- Término de processos
- Hierarquia de processos
- Estados de processos

Criação de processos



- Processos são criados
 - Na inicialização do sistema
 - A pedido do usuário
 - A partir de um processo em execução
 - A partir de uma tarefa em lote

- Processos de primeiro plano
- Processos de segundo plano
 - *Daemons*

- Processo pai cria um processo filho através de chamada de sistema
 - Processo filho é uma cópia do processo pai
 - Processo pai possui “ponteiro” para filho
 - Processo filho possui próprio espaço de endereçamento
 - Exceção: *copy-on-write*
 - Espaço de memória é compartilhado, mas modificações são copiadas para área privada de memória
 - Processos podem compartilhar recursos
 - Arquivos abertos
 - Permissões

Procedimento de criação de um processo



1. Definir um identificador de processo
2. Alocar espaço de memória para o processo
3. Criar uma entrada na tabela de processos
4. Definir o estado do processo
5. Criar (ou expandir) as estruturas de dados relativas

Término de processos



- Formas de término
 - Voluntária
 - Saída normal
 - Saída por erro
 - Involuntária
 - Erro fatal
 - Morte por outro processo

ATENÇÃO!!!!



- O Tanenbaum em português troca os erros
 - Erro fatal é confundido com saída por erro

- Programa termina execução e realiza chamada de sistema para informar o término
 - Libera recursos alocados
 - Memória
 - Arquivos abertos
 - Permissões de uso exclusivo
- Chamadas para saída
 - exit
 - UNIX
 - ExitProcess
 - Windows

Saída por erro



- Erro interno do processo afeta seu funcionamento
 - Avisa ao usuário
 - Termina
- Processos podem tentar corrigir o erro ao invés de fechar
 - Aviso na Interface Gráfica de Usuário (*Graphic User Interface* – GUI)

Erro fatal



- Processo é encerrado pelo operacional
 - Instrução ilegal
 - Acesso ilegal
- Operacional pode deixar o processo lidar com o erro
 - Operacional envia um sinal ao processo quando há erro
 - Depende do erro

Lista de erros fatais



- Vários tipos de erros fatais
 - Tempo excedido
 - Memória não disponível
 - Erro de proteção
 - Erro Aritmético
 - Espera excedida (*Time overrun*)
 - Falha de I/O
 - Instrução inválida
 - Instrução sem privilégios
 - Mal uso de dados

Morte por outro processo



- Outro processo realiza chamada ao operacional para terminar o processo atual
 - kill
 - UNIX
- Processo precisa de autorização para matar outro
 - Término do pai geralmente inicia término dos filhos

- *Exception vs Error*
 - Linguagens diferenciam entre os dois
 - Java
 - Haskell
 - C++
 - Mais ou menos
 - Exceção é um erro lógico interno
 - Frequentemente tratável
 - Erro é um problema que não é tolerado pelo operacional
 - Aplicação nem sempre pode tentar resolver
 - Pode ser considerado problema de segurança

Hierarquia de processos



- UNIX
 - Pais & filhos formam um grupo de processos
 - Processos pai repassam sinais aos filhos
 - Terminação do pai é avisada aos filhos
- Windows
 - Pais possuem “ponteiro” para filhos (*handle*)
 - Pais podem passar o “ponteiro” para outros processos
 - Quebra de hierarquia

Estados de processos



- Novo
 - Processo é inicializado
- Pronto
 - Processo está aguardando sua vez de ser executado
- Bloqueado
 - Processo está aguardando algum evento
 - Procedimento de I/O
- Executando
 - Processo está em execução
- Terminado
 - Processo terminou

Esta lista não
é exhaustiva!

Estados de processos

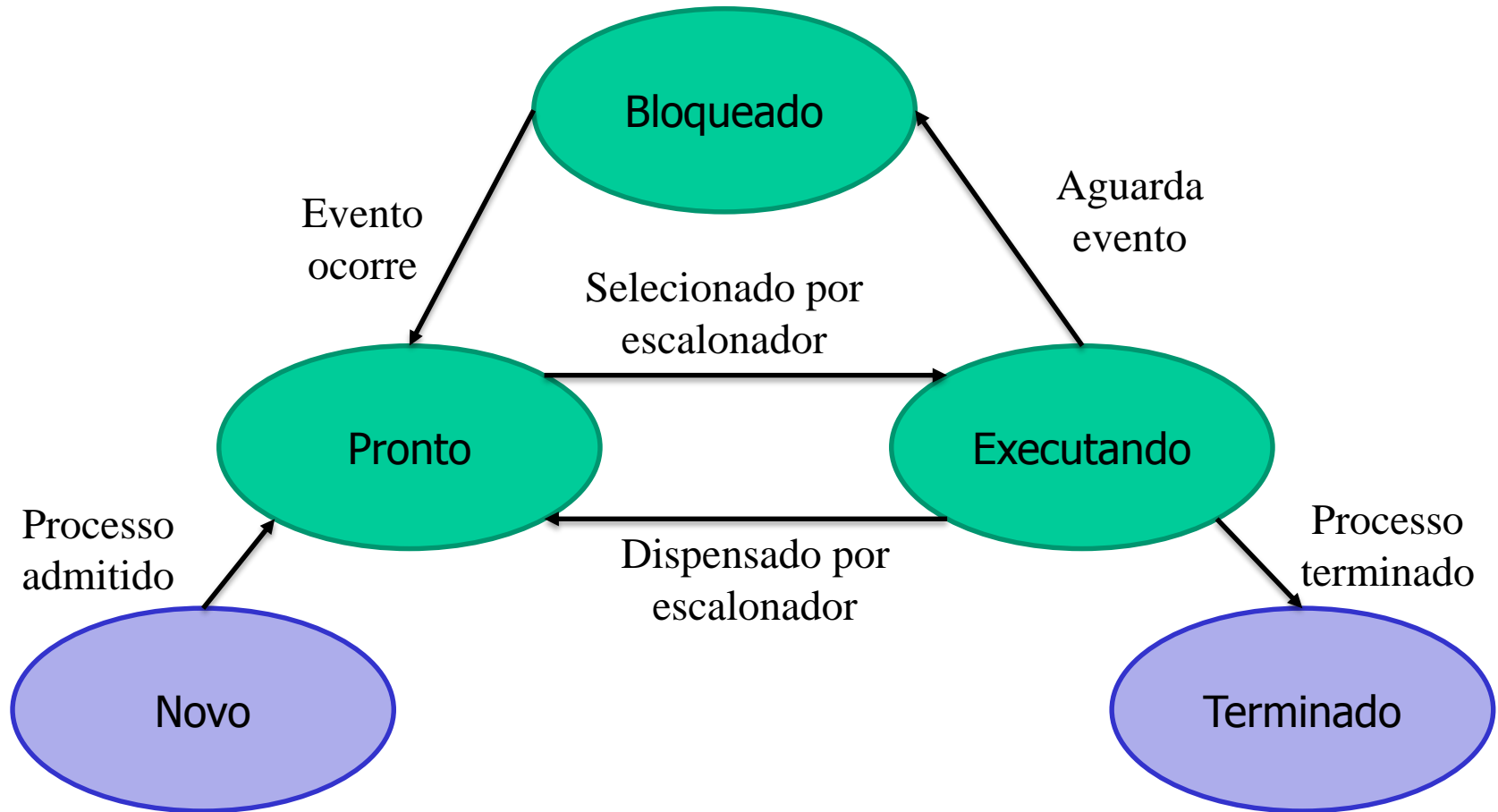


Tabela de processos – gerenciamento de processos



- Registros
- Contador de programa
- Palavra de estado do programa (PSW)
 - Conjunto de *flags*
- Ponteiro para a pilha
- Estado do processo
- Parâmetros de escalonamento
- Id do processo
- Prioridade
- Processo pai
- Grupo de processos
- Sinais
- Tempo de início
- Tempo de CPU
- Tempo de CPU de filhos

Tabela de processos – gerenciamento de memória



- Informações sobre segmento de código
- Informações sobre o segmento de dados
- Informações sobre o segmento de pilha

Tabela de processos – gerenciamento de arquivos



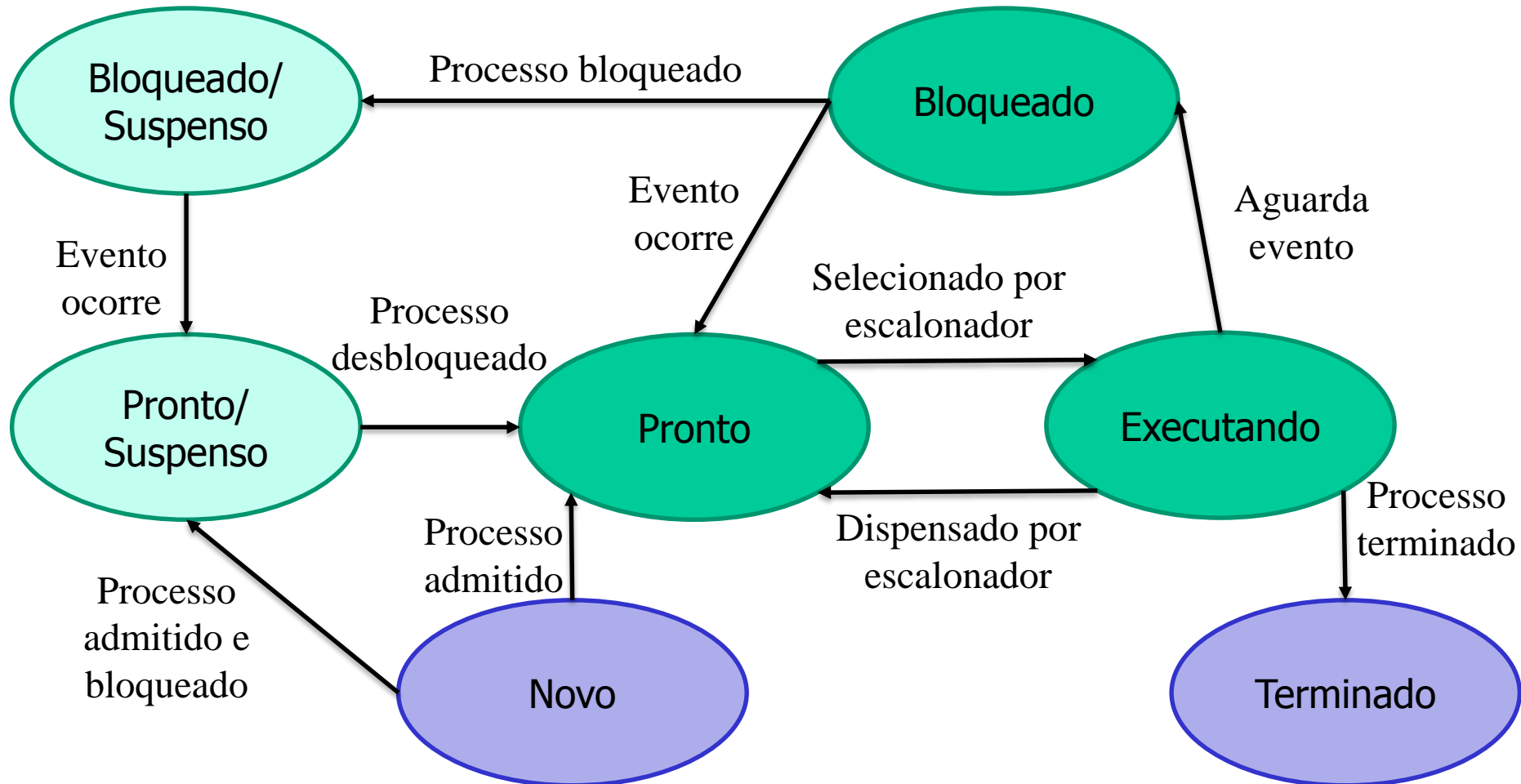
- Diretório-raiz
- Diretório corrente
- Descritores de arquivo
 - Ponteiros para *stdin*, *stdout*, *stderr*
- Id do usuário
- Id do grupo

Processos suspensos



- Tipicamente:
 - Vários processos podem estar aguardando operações de I/O
 - Bloqueados
 - Subutilização da CPU
 - Memória principal não comporta processos suficientes para encher a CPU de trabalho
 - I/O de disco é mais rápido que outras operações de I/O
- Processos podem ser guardados na memória secundária
 - Suspensos
- Intervalos de CPU estimulam a troca de processos entre as memórias

Estados de processos com suspensão



- Comando do Linux para gerenciar processos

```

gta@niteroi: ~
gta@niteroi: ~ 118x66
top - 12:07:56 up 2:20, 2 users, load average: 0,16, 0,17, 0,24
Tasks: 242 total, 1 running, 241 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0,7 us, 0,5 sy, 0,0 ni, 98,7 id, 0,1 wa, 0,0 hi, 0,0 si, 0,0 st
KiB Mem: 8117576 total, 5895752 used, 2221824 free, 216180 buffers
KiB Swap: 34185208 total, 0 used, 34185208 free, 623988 cached

```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
6727	gta	20	0	5527m	3,7g	3,7g	S	9	48,3	5:35.12	VirtualBox
4616	gta	20	0	2451m	234m	68m	S	6	3,0	10:52.59	Web Content
4386	gta	20	0	1685m	105m	36m	S	1	1,3	2:47.05	gnome-shell
4485	gta	20	0	2316m	256m	77m	S	1	3,2	4:24.49	firefox-esr
6700	gta	20	0	643m	11m	8272	S	1	0,1	0:08.24	VBoxSVC
3567	root	20	0	266m	51m	35m	S	1	0,6	1:37.01	Xorg
2745	mysql	20	0	547m	52m	7216	S	0	0,7	0:06.08	mysqld
4455	gta	20	0	5962m	147m	27m	S	0	1,9	0:24.48	dropbox
5643	root	20	0	0	0	0	S	0	0,0	0:03.30	kworker/4:2
6680	gta	20	0	856m	46m	31m	S	0	0,6	0:06.25	VirtualBox
6695	gta	20	0	117m	6800	4332	S	0	0,1	0:04.03	VBoxXPCOMIPCD
7000	gta	20	0	23596	1764	1144	R	0	0,0	0:00.11	top
1	root	20	0	15480	884	724	S	0	0,0	0:00.64	init
2	root	20	0	0	0	0	S	0	0,0	0:00.00	kthreadd
3	root	20	0	0	0	0	S	0	0,0	0:25.43	ksoftirqd/0
5	root	0	-20	0	0	0	S	0	0,0	0:00.00	kworker/0:0H
7	root	20	0	0	0	0	S	0	0,0	0:02.84	rcu_sched
8	root	20	0	0	0	0	S	0	0,0	0:00.00	rcu_bh

- Comando do Linux para gerenciar processos

```
gta@niteroi: ~  
gta@niteroi: ~ 118x66  
1 [|||] 3.8%] 4 [ 0.0%] 7 [|||] 3.4%] 10 [ 0.0%]  
2 [ 0.0%] 5 [|||] 8 [ 4.3%] 11 [|||] 1.9%]  
3 [|||] 2.4%] 6 [ 0.0%] 9 [|||] 4.7%] 12 [ 0.0%]  
Mem[|||||] 4958/7927MB] Tasks: 96, 397 thr, 145 kthr; 1 running  
Swp[ 0/33383MB] Load average: 0.42 0.28 0.27  
Uptime: 02:22:57  
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command  
6738 gta 20 0 5522M 3826M 3782M S 0.5 48.3 0:02.67 /usr/lib/virtualbox/VirtualBox --comment win7 --startvm  
6739 gta 20 0 5522M 3826M 3782M S 1.9 48.3 0:17.60 /usr/lib/virtualbox/VirtualBox --comment win7 --startvm  
6743 gta 20 0 5522M 3826M 3782M S 0.5 48.3 0:00.37 /usr/lib/virtualbox/VirtualBox --comment win7 --startvm  
6756 gta 20 0 5522M 3826M 3782M S 0.0 48.3 0:00.13 /usr/lib/virtualbox/VirtualBox --comment win7 --startvm  
6757 gta 20 0 5522M 3826M 3782M S 8.1 48.3 4:58.67 /usr/lib/virtualbox/VirtualBox --comment win7 --startvm  
6759 gta 20 0 5522M 3826M 3782M S 0.0 48.3 0:00.00 /usr/lib/virtualbox/VirtualBox --comment win7 --startvm  
6761 gta 20 0 5522M 3826M 3782M S 0.0 48.3 0:00.00 /usr/lib/virtualbox/VirtualBox --comment win7 --startvm  
6762 gta 20 0 5522M 3826M 3782M S 0.0 48.3 0:00.14 /usr/lib/virtualbox/VirtualBox --comment win7 --startvm  
6763 gta 20 0 5522M 3826M 3782M S 0.0 48.3 0:00.07 /usr/lib/virtualbox/VirtualBox --comment win7 --startvm  
6764 gta 20 0 5522M 3826M 3782M S 0.0 48.3 0:00.00 /usr/lib/virtualbox/VirtualBox --comment win7 --startvm  
6772 gta 20 0 5522M 3826M 3782M S 0.0 48.3 0:05.36 /usr/lib/virtualbox/VirtualBox --comment win7 --startvm  
6773 gta 20 0 5522M 3826M 3782M S 0.0 48.3 0:00.40 /usr/lib/virtualbox/VirtualBox --comment win7 --startvm  
6774 gta 20 0 5522M 3826M 3782M S 0.5 48.3 0:00.12 /usr/lib/virtualbox/VirtualBox --comment win7 --startvm  
6775 gta 20 0 5522M 3826M 3782M S 0.0 48.3 0:02.33 /usr/lib/virtualbox/VirtualBox --comment win7 --startvm  
6776 gta 20 0 5522M 3826M 3782M S 0.0 48.3 0:03.93 /usr/lib/virtualbox/VirtualBox --comment win7 --startvm  
6777 gta 20 0 5522M 3826M 3782M S 0.0 48.3 0:00.00 /usr/lib/virtualbox/VirtualBox --comment win7 --startvm  
6778 gta 20 0 5522M 3826M 3782M S 0.0 48.3 0:00.30 /usr/lib/virtualbox/VirtualBox --comment win7 --startvm  
6779 gta 20 0 5522M 3826M 3782M S 0.0 48.3 0:00.33 /usr/lib/virtualbox/VirtualBox --comment win7 --startvm  
6780 gta 20 0 5522M 3826M 3782M S 0.0 48.3 0:00.00 /usr/lib/virtualbox/VirtualBox --comment win7 --startvm
```

- Comando do Linux para saber os limites de recursos por processo

```
gta@niteroi:~$ ulimit -a
core file size          (blocks, -c) 0
data seg size          (kbytes, -d) unlimited
scheduling priority    (-e) 0
file size              (blocks, -f) unlimited
pending signals        (-i) 63293
max locked memory      (kbytes, -l) 64
max memory size        (kbytes, -m) unlimited
open files             (-n) 1024
pipe size              (512 bytes, -p) 8
POSIX message queues   (bytes, -q) 819200
real-time priority     (-r) 0
stack size             (kbytes, -s) 8192
cpu time               (seconds, -t) unlimited
max user processes     (-u) 63293
virtual memory         (kbytes, -v) unlimited
file locks             (-x) unlimited
gta@niteroi:~$
```

Processos

Pedro Cruz

EEL770 – Sistemas Operacionais