

```
root@debian-13:~# █
```

Domínio Absoluto sobre o Debian 13

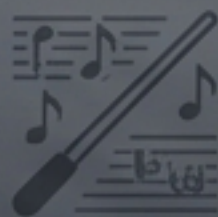
Arquitetura, Segurança e Gerenciamento de Serviços com systemd

O Roteiro da Sessão

**O Despertar:
Fluxo de Boot
& PID 1**



**O Maestro:
Fundamentos
do systemctl**



**O Blueprint:
Anatomia de
Unidades**



**A Caixa Preta:
Auditoria com
journalctl**



**O Crisol:
Laboratório
Prático**



UEFI/BIOS

Inicialização de hardware (POST)



GRUB

Carrega o Kernel na memória



Kernel Linux

Assume controle de CPU,
Memória e IPC



systemd (PID 1)

O primeiro processo do espaço
do usuário é despertado

O Paradoxo do PID 1

Um segundo após o boot, o Kernel delega a responsabilidade total da inicialização e orquestração do ecossistema de software ao systemd.

0 Kernel Linux vs 0 systemd

Ring 0

Ring 3

0 Kernel Linux (Modo Supervisor)

- ✓ Gerencia Escalonamento (CFS, FIFO)
- ✓ Gerencia Memória e Hardware
- ✓ Comunicação Inter Processos (IPC)

0 systemd (Modo Usuário)

- ✓ Gerencia Daemons e Serviços
- ✓ Resolve Dependências de Boot
- ✓ Gerencia Logs e Sockets

O Kernel fornece os recursos de hardware; o systemd orquestra como as aplicações os utilizam.

[Kernel]

[PID 1: systemd]

systemd-journald.service

Logs

sshd.service

Rede

cron.service

Agendamento

custom.service

Objetivo do Laboratório

Todo processo de usuário no Debian 13 descende direta ou indiretamente do PID 1. Se o PID 1 falha, ocorre um Kernel Panic.

systemctl: A Chave Mestra

`systemctl` [AÇÃO] [UNIDADE]

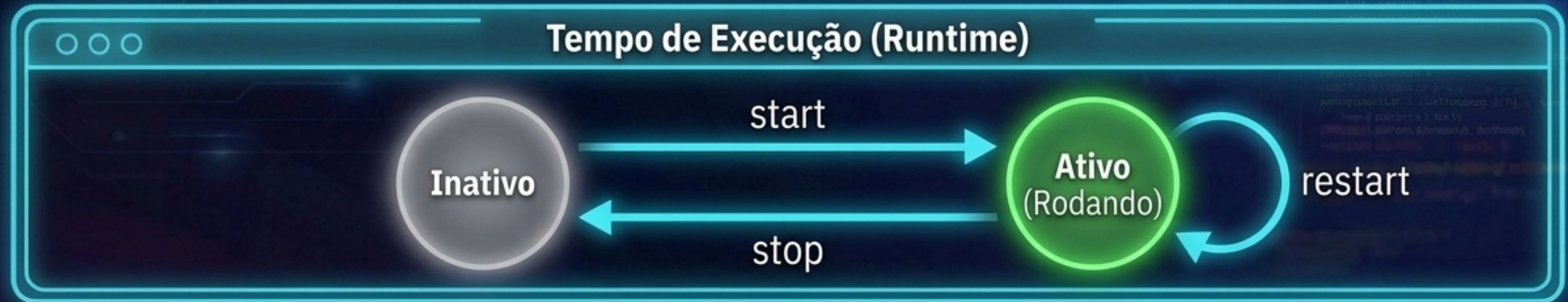
O que você quer fazer
(start, stop, status, restart)

O alvo da ação
(ex: ssh.service, apache2.service)



Pro-Tip: A extensão `.service` é o padrão.
Digitar `'systemctl status ssh'` é interpretado
como `'ssh.service'` automaticamente.

Service States: Runtime vs. Boot Time



Atenção: Um serviço pode estar Inativo agora, mas Enabled para o próximo boot!
Sempre verifique ambos os estados.



Evolução do Gerenciamento de Processos

Abordagem Legada (UNIX Clássico)

```
kill -9 PID / killall
```

```
ps aux | grep [nome]
```

```
nice -n 19
```

Abordagem Moderna (systemd)

```
systemctl kill [serviço]
```

Vantagem: mata todos os processos-filhos agrupados via cgroups, sem zumbis.

```
systemctl status [serviço]
```

Vantagem: Mostra logs recentes, PID principal e árvore de processos completa.

Diretiva **CPUSchedulingPolicy=no** .service

Onde Vivem as Unidades (Units)?



`/lib/systemd/system/`

Arquivos originais distribuídos pelos mantenedores de pacotes (Apt/Debian).

Arquivos originais distribuídos pelos mantenedores de pacotes (Apt/Debian).

REGRA: Nunca edite diretamente. Atualizações do sistema sobrescreverão suas mudanças.



`/etc/systemd/system/`

O domínio exclusivo do Administrador de Sistemas.

REGRA: Sempre crie e edite seus arquivos customizados aqui. O systemd prioriza este diretório.

Anatomia de um Arquivo de Serviço (systemd)

... Anatomia do Arquivo .service

[Unit]

Description=Daemon de Monitoramento
After=network.target

[Service]

ExecStart=/usr/local/bin/monitor.sh
Restart=on-failure

[Install]

WantedBy=multi-user.target

[Unit]

Define metadados e dependências. (Ex: Só inicie este serviço após a rede estar operante).

[Service]

O coração do arquivo. Define qual script executar e como reagir a falhas (Auto-healing).

[Install]

Onde plugar este serviço durante o fluxo de boot (equivalente aos runlevels clássicos).

Hardening: O Princípio do Menor Privilégio

Por que rodar um daemon como root se ele não precisa de acesso irrestrito ao sistema?



User=monitoruser

Executa o processo como um usuário sem privilégios de login (/bin/false).

ProtectSystem=strict

Monta toda a hierarquia do SO (/usr, /boot) como Somente Leitura para o daemon.

ProtectHome=yes

Torna todos os diretórios /home invisíveis e inacessíveis para o script.

A Evolução da Auditoria de Logs



`/var/log/messages` (Texto Puro)



`systemd-journald` (Binário Estruturado)

- Adeus comandos `grep` complexos e frágeis em arquivos de texto puro.
- O `journald` unifica `stdout`, `stderr`, `syslog` e logs do kernel em um único índice.
- Formato binário: Resistente a adulteração e injeção por atacantes.
- Pesquisa nativa, indexada e rápida baseada em metadados (PID, UID, Nome do Serviço).

O Funil do journalctl



Milhões de Logs
Brutos do Sistema

-u meu-servico
Isola a unidade

-p err
Isola a prioridade

--since '1 hour ago'
Isola o período

```
journalctl -u meu-servico -p err --since '1 hour ago' -f
```

(A flag -f mantém o log aberto em tempo real, semelhante ao tail -f)

Missão do Laboratório: O Daemon Resiliente

Objetivo: Criar um script de monitoramento de recursos, isolá-lo de privilégios de sistema, orquestrá-lo com um `.service`, e aplicar Chaos Engineering induzindo falhas para testar a resiliência.



Passo 1: O Payload e a Identidade

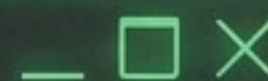


Criar o Usuário de Serviço

```
sudo useradd -r -s /bin/false monitoruser
```

(A flag -r cria uma conta de sistema, -s /bin/false impede login interativo no shell).

O Motor (Script Bash: /usr/local/bin/cpu-monitor.sh)



```
#!/bin/bash
while true; do
    echo 'Uso de CPU registrado às $(date)'
    sleep 5
done
```

```
sudo chmod +x /usr/local/bin/cpu-monitor.sh
```


Passo 2: Construindo o Blueprint



```
sudo nano /etc/systemd/system/cpu-monitor.service
```

```
[Unit]
```

```
Description=Monitor de CPU Seguro
```

```
[Service]
```

```
ExecStart=/usr/local/bin/cpu-monitor.sh
```

```
Restart=always
```

```
User=monitoruser
```

```
ProtectSystem=strict
```

```
[Install]
```

```
WantedBy=multi-user.target
```

Security Check

- ✓ Isola o processo
(User=monitoruser)
- ✓ Garante imortalidade
(Restart=always)
- ✓ Blinda o SO contra o script
(ProtectSystem=strict)

Passo 3: A Trindade da Implantação



Terminal Window 1

```
sudo systemctl daemon-reload
```

✓ Obrigatório: O systemd recarrega seus mapas internos e reconhece o novo arquivo recém-criado.

```
sudo systemctl enable cpu-monitor
```

✓ Garante a persistência: Criação do symlink para inicialização no próximo boot.

```
sudo systemctl start cpu-monitor  
sudo systemctl status cpu-monitor
```

✓ Acende os motores: Inicia o processo agora. Verifique o LED verde indicando 'active (running)'.

Passo 4: Chaos Engineering (Teste de Fogo)



Induzindo a Falha Catastrófica

- ✓ Encontre o PID atual:
`"systemctl status cpu-monitor"`
- ✓ Execute o assassinato:
`"sudo kill -9 [PID_DO_SCRIPT]"`

Auditoria e Auto-Cura

- ✓ Assista a magia ocorrer na caixa preta:
`"sudo journalctl -u cpu-monitor -f"`
- ✓ Resultado Esperado: O log registrará a morte violenta do processo e, em frações de segundo, o systemd irá ressuscitá-lo devido à diretiva **Restart=always**, gerando um novo PID automaticamente.

A Mudança de Paradigma Operacional



Da Anarquia à Ordem

Superamos os scripts complexos e soltos do SysVinit para adotar uma orquestração centralizada e declarativa.

Segurança por Design

Serviços modernos nascem em jaulas criptográficas isoladas (Least Privilege), não como instâncias irrestritas de root.

Imortalidade Gerenciada

O PID 1 não apenas inicia o sistema operacional; ele é o cão de guarda perpétuo e resiliente da sua infraestrutura.

O terminal é de vocês. O laboratório está oficialmente aberto.