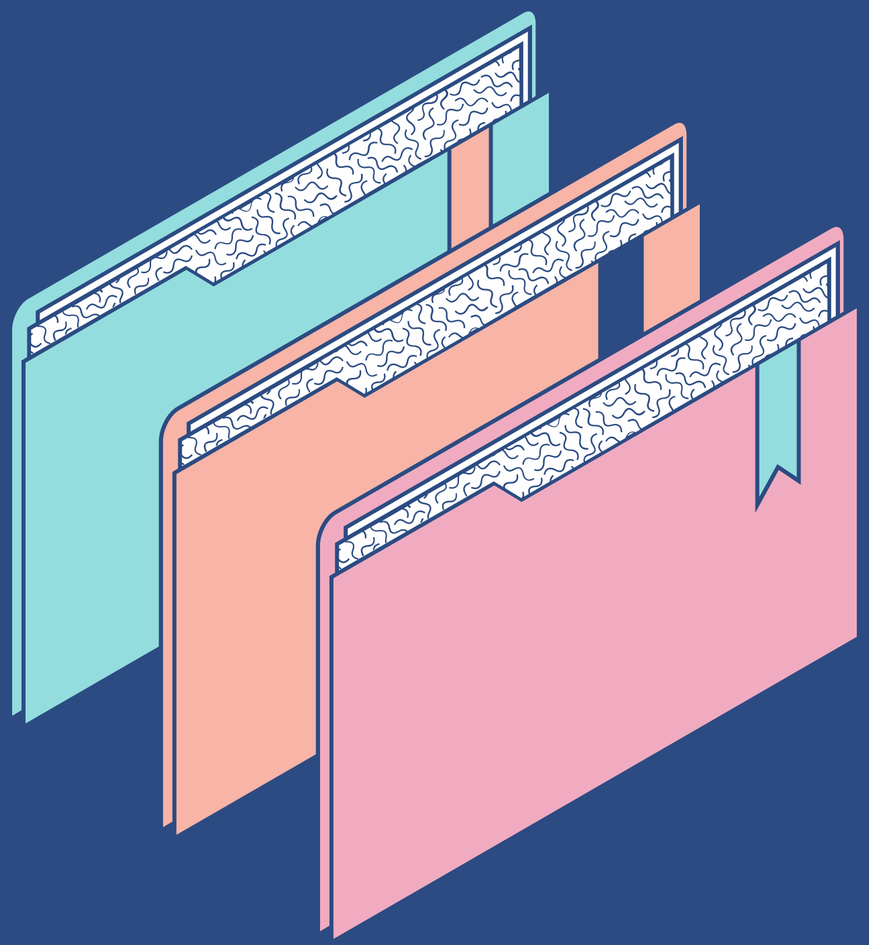
SCHEDULING POLICY TO THE LINUX KERNEL

Política de Escalonamento para o Kernel Linux

Eduardo H. A. Izidorio | Gabriel P. M. Costa



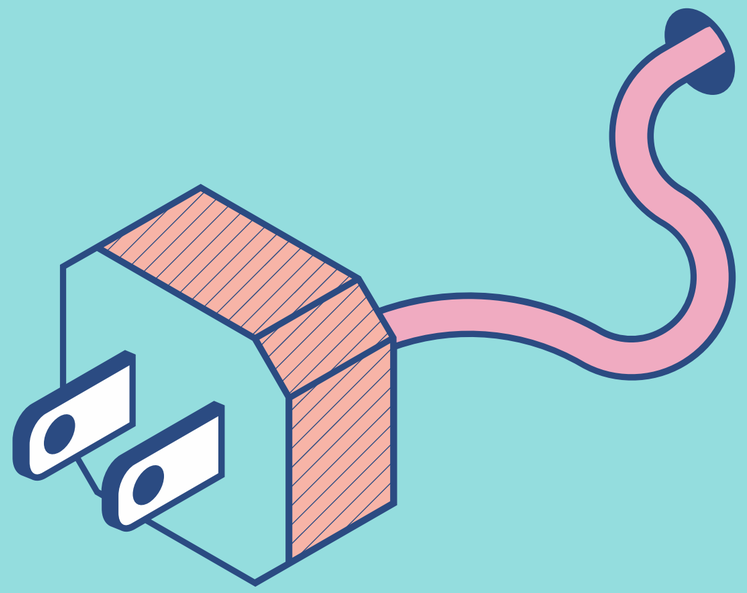
Tópicos

PRINCIPAIS TÓPICOS DISCUTIDOS NESTA APRESENTAÇÃO

- O que é o Escalonador do Linux?
- Como é o Funcionamento do Escalonador
- Estados das Threads
- Classificações de uma Thread
- Tempo Real e Não Real
- Políticas de Escalonamento do Linux
- Tutorial

O que é o Escalonador do Linux?

Em sistemas de computadores modernos, pode haver muitas threads esperando para serem atendidas ao mesmo tempo, assim como uma parte importante do Kernel, o escalonador, é responsável por decidir qual thread executa e por quanto tempo.

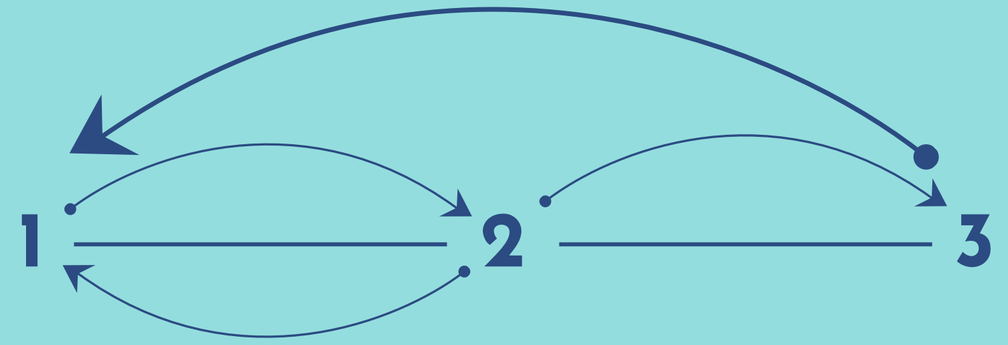


Como é o Funcionamento do Escalonador

- O kernel do Linux apresenta um agendamento preemptivo, isso significa que em determinada situação uma thread pode ser parada para execução de outra antes de ser concluída

- O escalanador do linux trabalha com bases em prioridades desta maneira as threads ou processos que possuírem uma maior prioridade são executados primeiro e os que possuírem níveis equivalentes de prioridade serão executados de acordo com a política round-robin de escalonamento. Com isso o encadeamento antecipado retoma a thread suspensa após a outra finalizar ou simplesmente ser bloqueada.

Estados das Threads



PRONTO

O encadeamento está pronto para ser executado, mas não tem permissão, ele só terá a permissão após os processadores desocuparem os que estão executando e se o escalonador o escolher;

EXECUTANDO

A thread está sendo executada em um processador;

BLOCKED

A thread está bloqueada esperando apenas algum evento externo como um I/O Bound ou um sinal. O thread bloqueado não pode ser executado.

Classificações de uma Thread

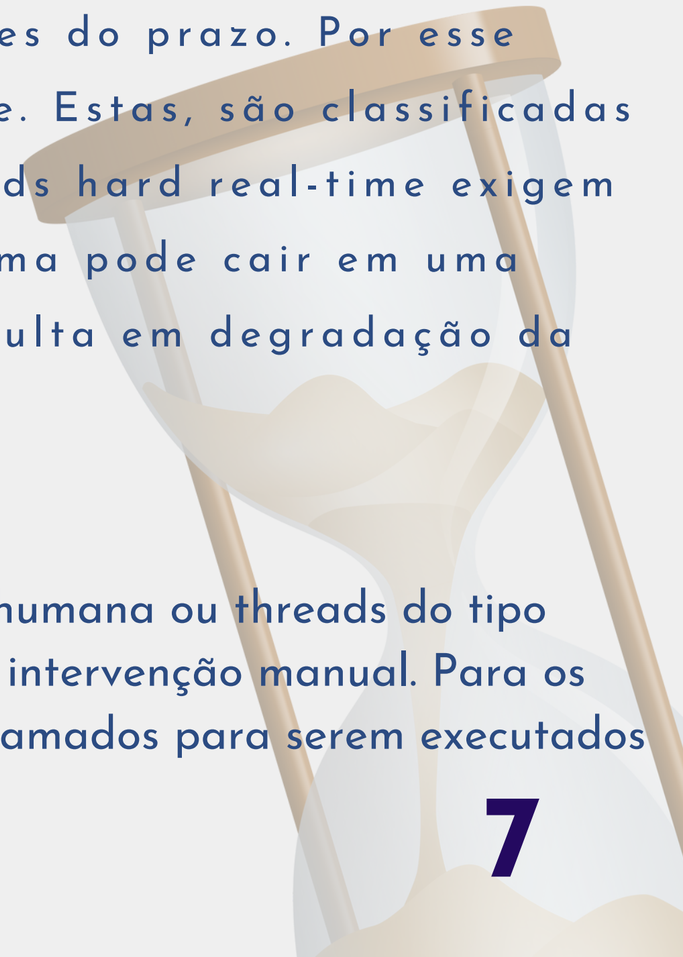
I/O-Bound: São encadeamentos que fazem o uso intensivo de chegadas de entradas(por exemplo, cliques no mouse ou até copiar um arquivo para um pendrive) ou a conclusão de saídas(por exemplo, gravação em discos). Esses processos são conhecidos assim por fazer pouco uso da CPU e por isso eles são processados mais rapidamente e o escalonador considera que ele tenha uma maior prioridade em relação ao CPU-Bound;

CPU-Bound: São aqueles onde o tempo de processamento depende mais do processador do que as entradas e saídas, fazendo assim com que atrapalhe o tempo total de processamento (como por exemplo,a compilação de programa). As threads que têm vinculação com o CPU são pouco escolhidas, mas em compensação quando são escolhidas elas mantêm a CPU por mais tempo.

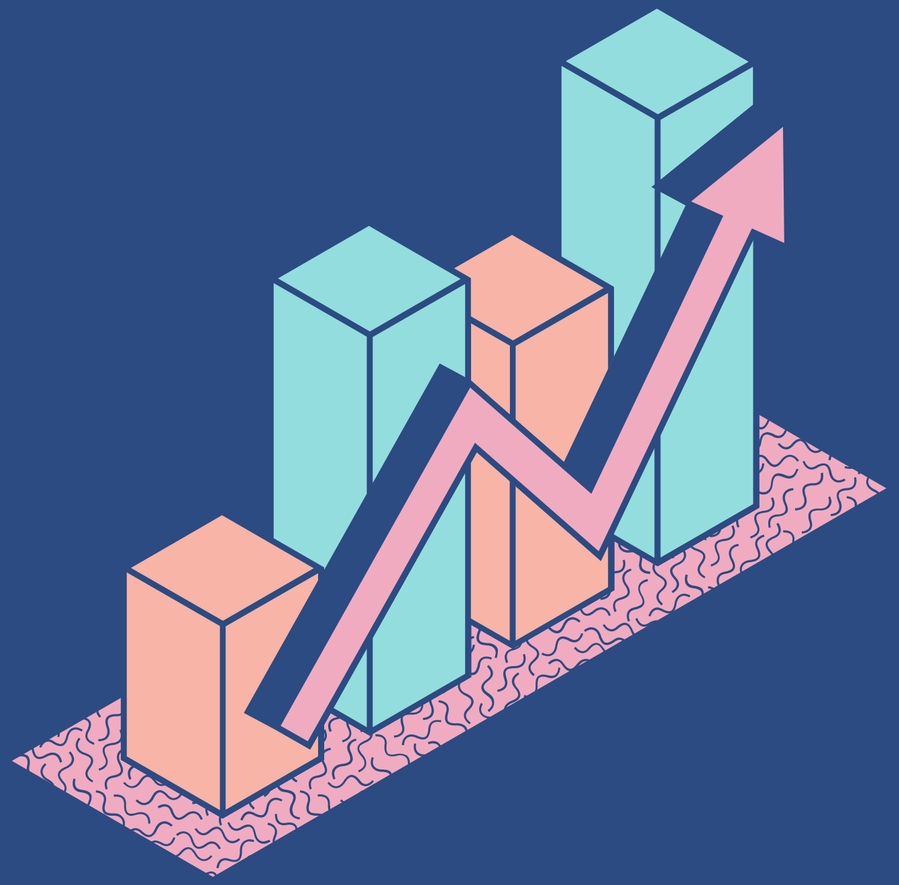
Tempo Real e Não Real

Real-time: são aquelas threads que geralmente trabalham com uma restrição de tempo, um prazo, também conhecido como deadline. A exatidão operacional de um encadeamento não depende apenas do resultado do cálculo, mas também de os resultados serem entregues antes do prazo. Por esse motivo o escalonador deixa com a mais alta prioridade as threads real-time. Estas, são classificadas como hard real-time ou soft real-time (devido à perda de um prazo), Threads hard real-time exigem que todos os prazos sejam cumpridos sem exceção, caso contrário, um sistema pode cair em uma falha catastrófica. Agora, para threads soft real-time, perder um prazo resulta em degradação da qualidade do serviço pretendido, mas o sistema ainda pode continuar.

Non-real-time: não está associado a nenhum prazo. Elas podem ser threads com interação humana ou threads do tipo batch, os threads batch são aquelas que processam uma grande quantidade de dados sem intervenção manual. Para os batch threads, um tempo de resposta rápido não é crítico e, portanto, eles podem ser programados para serem executados conforme os recursos permitirem.



Políticas de Escalonamento do Linux



First Come, First Served(FCFS)

FIRST IN, FIRST OUT(FIFO)

É um algoritmo de escalonamento não preemptivo para a estruturas de dados do tipo fila. E tem como principal conceito: “o primeiro elemento a dar entrada na fila é o que será processado primeiro”.
Isso quer dizer que ele executará um processo como todo do início ao fim não interrompendo o processo executado até ser finalizado, então se chegar um novo processo e estiver algo processando esse novo processo vai para a fila de espera até ser atendido pela CPU.

ROUND-ROBIN

ESCALONAMENTO POR REVEZAMENTO

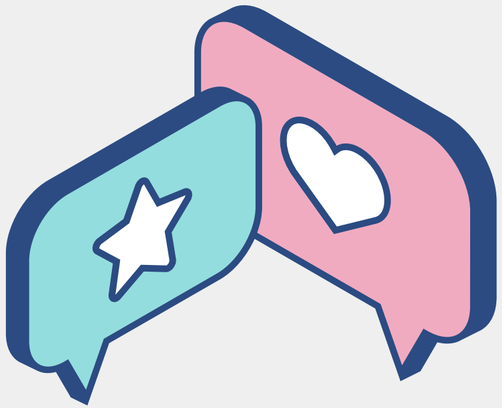
É um algoritmo escalonador de tarefas que consiste em dividir o tempo de uso da CPU. Cada processo recebe uma fatia de tempo, esse tempo é chamado Time-Slice, também conhecido por Quantum. Os processos são todos armazenados em Fila (ou Buffer) circular.

O escalonador executa cada tarefa por tempo determinado pelo Time-Slice e ao fim deste período é executada a troca de contexto, onde o próximo processo da fila passa a ser executado pela CPU até percorrer o período do Time-Slice. Após percorrer todos os processos da fila, essas atividades se repetem e o escalonador aponta para a primeira tarefa.

Shortest Remaining Time First(SRTF)

MENOR TEMPO RESTANTE PRIMEIRO

Esse tipo de algoritmo é cooperativo, ou seja, uma vez que uma tarefa recebe o processador, ela executa até encerrar (ou liberá-lo explicitamente). Em uma variante preemptiva, o escalonador deve comparar a duração prevista de cada nova tarefa que ingressa no sistema com o tempo de processamento restante das demais tarefas presentes, inclusive aquela que está executando no momento. Caso a nova tarefa tenha um tempo restante menor, ela recebe o processador.



Escalonamento por prioridades fixas (PRIOc, PRIOp)

- PRIOc:

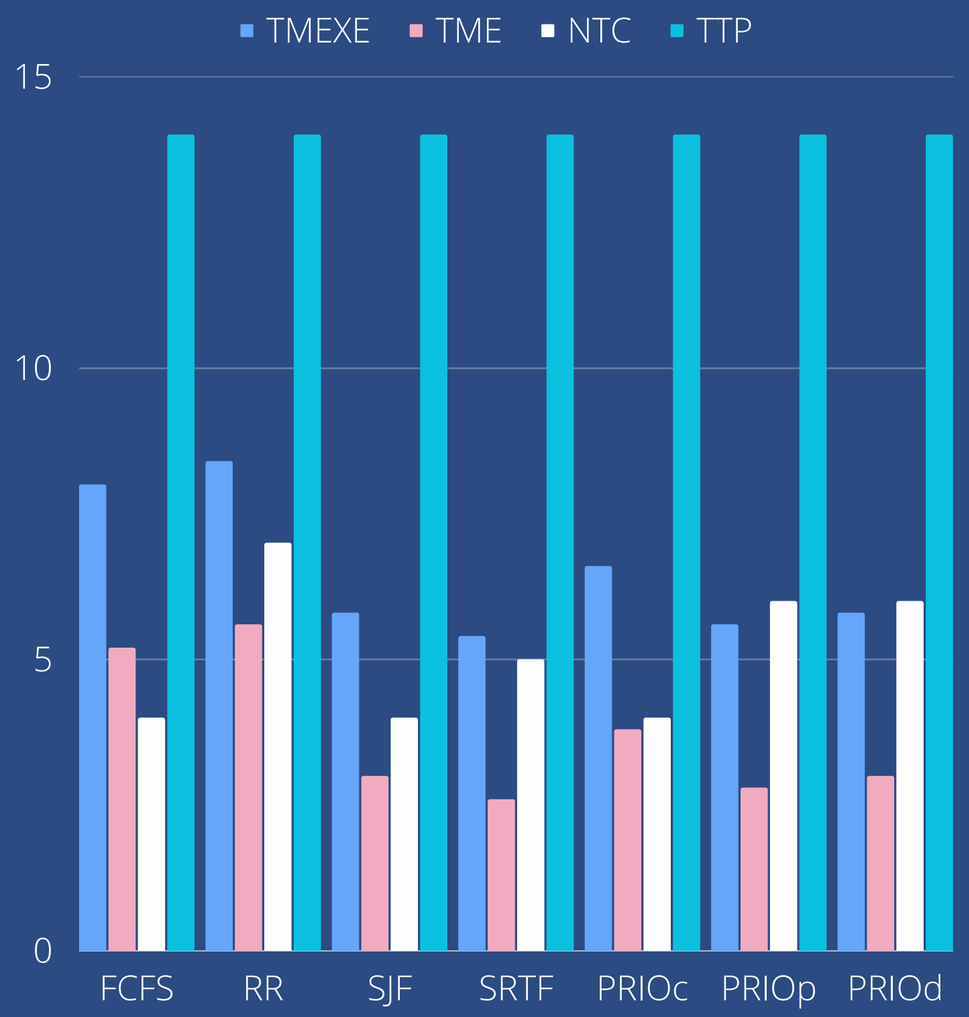
Os valores de prioridade são considerados em uma escala de prioridade positiva, ou seja, valores numéricos maiores indicam maior prioridade.

- PRIOp:

No escalonamento por prioridade preemptivo (PRIOp), quando uma tarefa de maior prioridade se torna disponível para execução, o escalonador entrega o processador a ela, trazendo a tarefa atualmente em execução de volta para a fila de prontas. Em outras palavras, a tarefa em execução pode ser "preemptada" por uma nova tarefa mais prioritária.

Comparação entre os algoritmos de Escalonamento

- Tempo médio de execução (TMEXE);
- Tempo médio de espera (TME);
- Número de trocas de contexto (NTC);
- Tempo total de processamento (TTP).



O Linux 2.6.24 é a versão padrão do kernel no Ubuntu 8.04, mas em versões mais recentes, a encapsulação é aumentada, o que aumenta a complexidade do código do agendador. Nas versões mais recentes, não há mais um arquivo "sched.c", mas sim um diretório "sched" que consiste em várias partes diferentes do antigo agendador. Por isso que antes de começar o tutorial precisamos dizer que é complexo demais fazer o escalonador nas versões mais atuais do kernel do linux

Tutorial

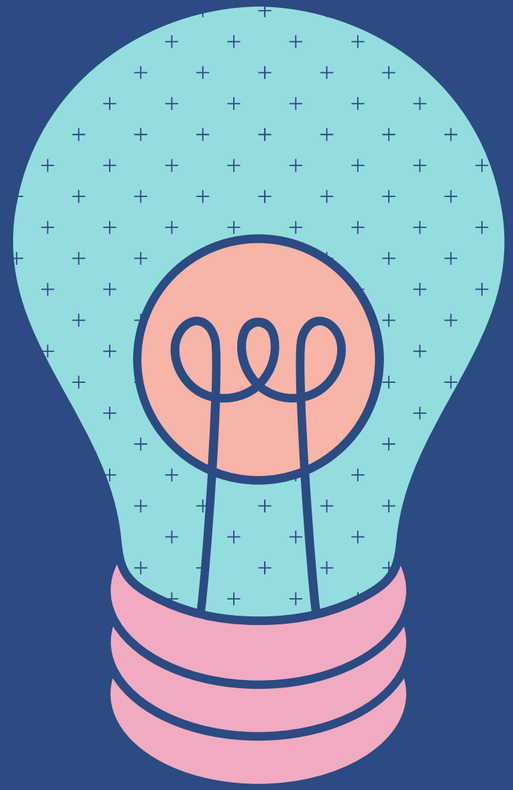
1. Para começar, precisamos descobrir qual versão do kernel estamos executando no momento:
$ uname -r

2. Obtenha o código do kernel Linux no site Kernel.org e usando o comando wget:
$ cd /tmp
$ wget https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux.git/snapshot/linux-2.6.24.y.tar.gz

3. Extraia o arquivo .tar:
$ tar -xzvf 2.6.24.y.tar.gz -C /usr/src
$ cd /usr/src/2.6.24.y

4. instalar a biblioteca curses e algumas outras ferramentas para nos ajudar a compilar :
$ sudo apt-get install kernel-package libncurses5-dev fakeroot

5. Aplicar as mudanças de implementação nas pastas sched.c,sched.h e chrtB.c:
/include/linux/sched.h
/kernel/sched.c
/usr/bin/chrt



6. Fazemos uma cópia da configuração do kernel para usar durante a compilação:
$ cp boot/config-2.6.24-generic /usr/src/linux/.config

7. Primeiro vamos fazer um make clean, só para ter certeza que está tudo pronto para a compilação:
$ make-kpkg clean

8. Em seguida, vamos realmente compilar o kernel. Isso levará um “TEMPO LONGO” talvez 40-50 minutos.
$ fakeroot make-kpkg --initrd --append-to-version=-custom kernel_image kernel_headers
Este processo criará dois arquivos .deb em /usr/src que contém o kernel

9.Observe que, ao executar os próximos comandos, isso definirá o novo kernel como o novo kernel padrão. Isso pode dar problema! Se sua máquina não inicializar, você pode pressionar Esc no menu de carregamento do GRUB e selecionar seu kernel antigo. Você pode então desabilitar o kernel em /boot/grub/menu.lst ou tentar compilar novamente.
$ dpkg -i linux-image-2.6.24-custom_2.6.24-custom-10.00.Custom_i386.deb
$ dpkg -i linux-headers-2.6.24-custom_2.6.24-custom-10.00.Custom_i386.deb

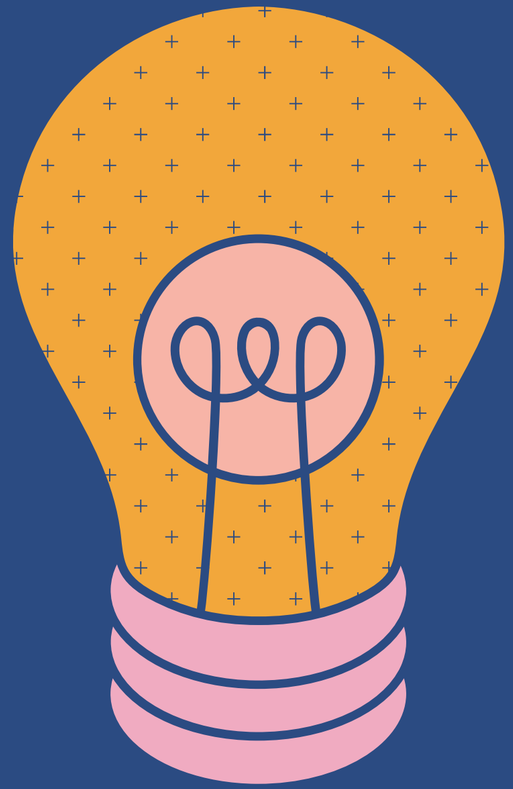
10. Agora reinicie sua máquina. Se tudo funcionar, você deve estar executando seu novo kernel personalizado. Você pode verificar isso usando uname. Observe que o número exato será diferente em sua máquina.:
$ uname -r
2.6.24-custom

11. Para alterar a política de um processo para SCHED_BACKGROUND política em seu tempo de execução, comando usado: :
$ chrt -g -p 0 < pid>

Tentativas

RTAI

PREEMPT-RT



OBRIGADO!

Eduardo H. A. Izidorio | Gabriel P. M. Costa