UNIX: processos e o kernel mode

O Kernel é um programa especial (Não é um processo), uma parte privilegiada do sistema operacional, responsável por funções críticas do S.O, como escalonamento da CPU e tratamento de interrupções.

Os processos têm ciclos de vida, criados pelas chamadas de sistemas fork ou vfork e terminadas pela primitiva exit. No Unix o processos possuem uma hierarquia, cada processo tem um processo pai (parent process) e pode ter um ou mais processos filhos (child processes).

O primeiro processo de usuário criado quando o sistema é iniciado, é chamado de init (PID 1) e se localiza no topo da hierarquia e todos processos dependem dele, com exceção dos processos de swapper ou o kthreadd (Linux), pois esses processos são implementados dentro do próprio kernel, ou seja, não há um programa (arquivo) binário regular para eles.

System(Kernel) Space x Process Space

O System Space e o Process Space são separados em diferentes pilhas, uma de karnel e outra de usuário, quando há chamadas a rotinas de usuário, é usada a pilha de usuário e quando há Chamadas de Sistema ou desvios devido a Interrupções de HW, é usada a pilha de Kernel, Isso garante mais segurança.

Estruturas de Controle do Unix

O “Bloco de Controle” no Unix é dividido em 2 partes:

- *Proc Structure*: Contém todas as informações que o kernel possa precisar quando um processo NÃO está running, mesmo que o processo vá para disco (susppended), sua Proc Struct permanece em memória! e se encontra no system (kernel) space.
- *u Área:* Contém informações necessárias apenas quando o processo está running, ou vai entrar nesse estado e se encontra no process space

User Mode x Kernel Mode

Com finalidade principal de proteção, a maioria das CPUs atuais fornece pelo menos dois modos de operação, no caso do Unix, requer do hardware a implementação de apenas 2 modos de operação:

User mode (menos privilegiado)

Os processos de usuários rodam em user mode, pois os processos não podem, de maneira acidental ou maliciosa, corromper outro processo ou até mesmo o kernel.

kernel mode (com mais privilégios)

O processo só consegue acessar o kernel space via SVC (portanto, acesso controlado).Existem 3 tipos de eventos que fazem o sistema (a CPU) entrar em kernel mode:

Cada tipo de interrupção está associado a um Interrupt Priority Level - IPL, e elas podem ocorrer enquanto uma outra está sendo atendida.

Chamadas ao sistema (SVCs/traps/interrupções de SW)

São eventos síncronos ao processo em execução e ocorrem quando o processo executa uma instrução especial como syscall ou trap, ele pode acessar o espaço de endereçamento do processo (incluindo u area), além de poder bloquear o processo, se necessário.

Interrupções de HW

São eventos gerados assincronamente à atividade regular do sistema, causados por dispositivos periféricos como relogio e disco, O processo que está rodando, passa de “user running” para “kernel running”, mas o código de kernel que passa a ser executado não tem nenhuma relação com o processo em si.

O nome dado à rotina de tratamento da interrupção é “Interrupt Handler” e ela roda em karnel mode, nessa rotina são executadas tarefas do kernel (de gerência do sistema) que não têm relação com o processo que se encontra “running” naquele instante,apesar disso, o tempo de servir a interrupção é descontado do quantum do processo em execução (time-slice).

Exceções

São eventos síncronos ao processo em execução, causados por eventos relacionados ao próprio processo, como a divisão por zero e overflow, são ditos síncronos pois, se e executarmos um mesmo código com as mesmas condições e dados de entradas, as mesmas exceções serão causadas.

Na ocorrência de uma delas, uma sequência especial de instruções é executada para colocar a CPU em kernel mode e passar o controle para o kernel (desvio para o código de kernel), o HW alterna para kernel mode utilizando a kernel stack do processo, salvando o a PC e Status (e possivelmente outros “estados” na kernel stack) e o SW, salva as outras informações necessárias para seguir o tratamento do evento.

Requisitos do Escalonador para cada tipo de Processo

Processos interativos

São interações que devem ser processadas rapidamente como editores e interfaces gráficas, é necessário ter um requisito para reduzir o tempo de resposta médio e sua variância, para que o usuário não perceba o atraso.

Processos batch

São processos que rodam em background sem a interação com o usuário,e tem como requisito, o tempo para completar uma tarefa na presença de outras atividades, comparado com turnaround (tempo para completar uma tarefa em um sistema “inativo”, sem outras atividades paralelas) deve ser razoável.

Processos de tempo real (soft)

São aplicações em time-critical, e tem como requisito que o seu escalonador tenha um comportamento previsível, com limites e tempos de resposta garantidos.

Sleep Priority

É uma prioridade do kernel, quando o processo está bloqueado, ele é associado a uma sleep priority relacionada com o dispositivo pelo qual ele está esperando e quando ele é acordado, de forma temporária, o processo será escalonado na frente de outros processos de usuário e continuará a sua execução em modo kernel a partir do ponto de bloqueio.Quando a SVC é finalmente completada, imediatamente antes de retornar ao modo usuário, o kernel recupera a prioridade normal do processo.

Escalonamento tradicional

O kernel do Unix tradicional é não-preemptivo, a chegada de um processo de mais alta prioridade na fila de prontos força a preempção do processo em execução SOMENTE SE ele está executando em user mode,

Um processo executando em kernel mode nunca é preemptado (nunca perde a posse da CPU para um outro processo naquele momento) e quando o sistema for retornar para user mode, ou seja, quando o que estava sendo executado em kernel mode (por exemplo, uma chamada de sistema) foi finalizado, aí sim ele pode ser preemptado, assim a chegada de um processo de mais alta prioridade na fila de prontos NÃO força a preempção imediata do processo que está rodando, Isso traz mais segurança ao SÓ já que com isso, a execução de uma rotina de kernel (por exemplo, uma SVC) não é interrompida e a rotina é executada até o final, quando o sistema for retornar para user mode, é verificado se há um processo mais prioritário na fila de prontos. Se sim, ocorre a preempção (troca de contexto).

Principal Problema com o Escalonador tradicional

Como o kernel é não-preemptivo, processos de maior prioridade podem ter que esperar muito tempo para ganhar a posse da CPU, não existindo garantias de tempos de resposta para aplicações com característica de tempo-real.

Kernel preemptivo - Implementação no SVR4

Processos de tempo real exigem tempos de resposta limitados, para isso, Preemption points são definidos em diferentes pontos do código do kernel, onde todas as estruturas de dados do kernel encontram-se estáveis ou onde kernel esteja prestes a iniciar alguma computação longa, em cada preemption point o kernel verifica se um processo de tempo-real tornou-se pronto e precisa ser executado, tomando como base os limites nos tempos máximos que um processo de tempo-real precisa esperar, esses tempos são definidos pelo maior intervalo entre dois preemption points consecutivos.

Algumas perguntas referente a aula:

Descreva de forma geral como funciona o Escalonamento Tradicional no UNIX.

Resposta:

O kernel do Unix tradicional funciona de modo não-preemptivo, a chegada de um processo de mais alta prioridade na fila de prontos força a preempção do processo em execução SOMENTE SE ele está executando em user mode,assim um processo executando em kernel mode nunca perde a posse da CPU para um outro processo naquele

momento e quando o sistema for retornar para user mode, ou seja, quando o que estava sendo executado em kernel mode foi finalizado, o processo retorna para user mode e é verificado se há um processo mais prioritário na fila de prontos. Se sim, ocorre a preempção (troca de contexto).

Em algumas implementações do UNIX, o kernel é não-preemptivo. O que isto significa? Quais as vantagens e desvantagens desta abordagem?

Resposta:

Significa que um processo executando em kernel mode nunca perde a posse da CPU para um outro processo naquele momento e só ocorre a troca de contexto quando o que estava sendo executado em kernel mode for finalizado, e se ao retornar para user mode, for encontrado um processo mais prioritário na fila de prontos, por não força a preempção imediata do processo que está rodando, há um aumento na segurança do Sistema Operacional, entretanto, como o kernel é não-preemptivo, processos de maior prioridade podem ter que esperar muito tempo para ganhar a posse da CPU, não existindo garantias de tempos de resposta para aplicações com característica de tempo-real.

Qual a diferença entre "Escalonamento Preemptivo" e "Kernel Preemptivo"?

Resposta:

No escalonamento preemptivo, a posse da CPU pode ser tomada do processo em execução a qualquer momento, já no Kernel preemptivo, um processo ganha posse dá CPU tomando como base os limites dos tempos máximos que um processo de tempo-real precisa esperar, e esses tempos são definidos pelo maior intervalo entre dois preemption points consecutivos.

No UNIX, um processo pode encontrar-se no estado Kernel Running devido a 3 eventos diferentes. Explique quais são eles

Resposta:

Chamadas ao sistema (SVCs/traps/interrupções de SW)

São eventos síncronos ao processo em execução e ocorrem quando o processo executa uma instrução especial como syscall ou trap, ele pode acessar o espaço de endereçamento do processo (incluindo u area), além de poder bloquear o processo, se necessário.

Interrupções de HW

São eventos gerados assincronamente à atividade regular do sistema, causados por dispositivos periféricos como relógio e disco, O processo que está rodando, passa de "user running" para "kernel running", mas o código de kernel que passa a ser executado não tem nenhuma relação com o processo em si.

Exceções

São eventos síncronos ao processo em execução, causados por eventos relacionados ao próprio processo, como a divisão por zero e overflow, são ditos síncronos pois, se e executarmos um mesmo código com as mesmas condições e dados de entradas, as mesmas exceções serão causadas.

Como se dá o processamento de uma interrupção de HW? Qual a relação entre interrupção e multiprogramação?

Resposta:

Quando ocorre uma interrupção, há desvio para o kernel, a interrupção passa a se tratada através rotina "Interrupt Handler" e ela roda em karnel mode, nessa rotina são executadas tarefas do kernel (de gerência do sistema) que não têm relação com o processo que se encontra "running" naquele instante,apesar disso, o tempo de servir a interrupção é descontado do quantum do processo em execução (time-slice), a relação com a multiprogramação está justamente nisso, enquanto o kernel trata da interrupção, através da multiprogramação, ao invés da CPU ficar ociosa, outro programa poderia ser processado, mantendo mais de um programa em execução simultaneamente, e tirando assim, o máximo de proveito do processador.

Qual a finalidade desses dois estados? Explique em que situações um processo passa do estado User-Running para o estado Kernel Running, e vice-versa.

Resposta:

User running é o estado menos privilegiado, usado para rodar aplicações de usuário, tem acesso restrito à memória. Kernel running é o estado mais privilegiado, acesso irrestrito à memória.Existem três situações em que o user running passa para o kernel running: houve uma interrupção de hardware, houve uma exceção ou o processo fez uma chamada de sistema. Após essas situações serem tratadas, o processo volta para user running.

Sempre que um processo é escalonado para rodar ele começa sua execução no estado Kernel-Running (inclusive pela 1a. vez que ele executa). Explique por que isso ocorre

Resposta:

Quando um processo "ganha a posse" da CPU pela primeira vez, ele começa rodando no estado "Kernel running", e em seguida passa para "User running" para executar código de usuário e outro processo ser escalonado, o primeiro processo estará no estado "Kernel Running", porque pode ser que tenha ocorrido uma interrupção de HW, exceção ou chamada e sistema