Multiprogramação: Conceitos de processos e threads

1. Processo: conceito mais central em S.O.
   1. Abstração de programa em execução;
   2. Operações (pseudo) concorrentes, mesmo com apenas um CPU disponível;
   3. Transforma uma única CPU em múltiplas CPUs virtuais.
2. Exemplo: Programa acesso o disco que dura uma eternidade, outro ode assumir a CPU.
3. CPU chaveia de programa para programa
   1. Cada um com dezenas ou centenas de milissegundos.
   2. A cada segundo, executa diversos programas.
      1. Pseudo Paralelismo X Paralelismo verdadeiro em multiprocessadores
4. Modelo de processo:
   1. Um programa em execução;
   2. Acompanhado dos valores atuais do contador de programa, registradores e variáveis (slide2).
5. Conceitualmente, cada processo tem sua CPU virtual.
6. Execução não é uniforme e nem reproduzível.
7. Diferença entre programa e processo
   1. Ex: receita de Bolo possui entrada, saída e estado.
   2. Programa executado 2 vezes constitui 2 processo.
      1. Código de programa pode ser compartilhado.
8. Criação de processo (silde3)
9. Termino de processos (silde4).
10. Hierarquia de processos
    1. Processo pai e processo filho ficam associados, criando um gruo de processos (unix).
    2. Windows não possui hierarquia, mas recebe handle, que pode repassar.
    3. Processos no unix não podem deserdar seus filhos.
11. Estados de processos (side 5)
12. Implementação de processos
    1. Tabela de processo (uma entrada para cada)
       1. (process control blocks)
          1. (Slides 6 a 9)
13. Threads
    1. Em SO´s tradicionais cada processo tem um espaço de endereçamento e um único thread de controle.
    2. Pode ser desejável situações com múltiplos threads de controle em um único espaço de endereçamento.
    3. Em muitas aplicações, ocorrem múltiplas atividades ao mesmo tempo;
    4. Dados são compartilhados entre theads do mesmo processo;
    5. Mais rápido criar e destruir threads do mesmo processo; (até 100x);
    6. Pode dividir E/S e execução na CPU em threads diferentes
    7. Em múltiplos CPU´s, paralelismo real no mesmo programa é possível (Ex: Planilha: interação, cálculo, backup) (slide 10).
14. Modelo de thread clássico
    1. Processo: meio de agrupar recurso (espaço de endereçamento, código, variáveis, arquivos abertos)
    2. Thread (processos leves): contador de programa que mantém o controle de qual instrução deve executar sem seguida (múltiplos threads em um único processo).
    3. Sistema multithread: Permite Múltiplas thread no mesmo processo (Slide 13)
    4. Não há proteção entre threads
       1. Processos são hotis, de múltiplos usuários, competem por recursos.
       2. Threads colaboram para executar uma tarefa.
    5. Threads posix
       1. Padrão IEEE 1003.1c (pthreads) (slide 16)
    6. Implementação de threads (slides 18 e 19)
       1. No espaço do usuário (núcleo não conhece threads)
          1. Algoritmos de escalonamento distintos
          2. Chamadas bloqueantes bloqueiam todas as threads (E programas em gera usam threads para evitar isso.
       2. No núcleo
          1. Custo maior de criação e destruição
          2. Uma thread pode bloquear. S.O. executa outra
       3. Híbridas
    7. Convertendo código monothread para multithread
       1. Problemas: Rotinas podem possuir variáveis globais (ex: Errno)
       2. Rotinas nem sempre são reentrantes.
       3. Tratamento de sinais.
       4. Gerenciamento de Pilha.