Aula 21

Proteção

Proteção – Objetivos

- manter a integridade do SO
- proteger de usuários bizantinos
- proteger de usuários incompetentes
- aumenta a confiabilidade detectando erros de interface

Proteção – Domínios de Proteção

- Computador: uma coleção de objetos, p.ex. CPU, arquivo, impressora, semáforos etc.
- Cada objeto tem um tipo de acesso diferente: **tipo abstrato de dados.**
- Processo deve ter acesso somente aos objetos de que precisa na hora em que for preciso
- Um domínio de proteção especifica quais objetos um processo pode acessar:
  - É uma coleção de pares ordenados
    <objeto, permissão>
    - exemplo: <arquivo_A, rw>

Proteção – Domínios de Proteção

- Domínios de proteção podem ter sobreposição;

- Podem ser atribuídos:
  - estaticamente
  - dinamicamente

Diferenças? Vantagens?

Proteção – Domínios de Proteção

- Tipos de domínios:
  - Usuário: p.ex. diretório, impressora, etc
  - Processo: p.ex. espaço de endereçamento
  - Procedimento: p.ex. variáveis locais

Proteção – Exemplos de Domínios de Proteção

- Unix:
  - domínio é por usuário
  - quando é necessário acessar objetos com permissões diferentes, **troca-se o usuário:** SETUID.
    - Problema!
    - Mas se não puder trocar o usuário, o sistema fica restritivo demais

Proteção – Exemplos de Domínios de Proteção

- Multics:
  - anéis de proteção:
    - cada anel é um domínio de proteção
    - anéis vão de 0 a 7, onde 0 tem maior acesso
    - anéis maiores são contidos em anéis menores.
  - modelo simplificado: dois anéis, 0: modo superusuário, 1: modo usuário
  - Sistema segmentado, cada segmento pertence a um anel
  - Além disso, cada segmento tem 3 bits rwx
  - Troca de domínio acontece quando um processo executando em um anel chama outro em um anel diferente

Proteção – Exemplos de Domínios de Proteção

- Multics:
  - para garantir proteção o segmento inclui:
    - access bracket: [b1, b2], Limite: b3
    - lista de portas de acesso limitado.
  - se um processo no anel i chama outro no anel j:
    - se i<b1, a chamada é permitida (i tem mais prioridade), mas parâmetros que se referem ao anel i são **copiados**.
    - a chamada é permitida se b1 <= i <= b2. Anel permanece i.
    - se b2 <= i <= b3, a chamada é permitida se for feita a uma das portas de acesso limitado.
    - senão a chamada é recusada.
  - desvantagens: complexo, proteção estática

Proteção – Matriz de Acesso

- Especifica quem pode acessar o quê:

	Arq1	*Arq2*	*Arq3*	*Impr.*
D1	r		r	
D2				u
D3		r	x	
D4	rw		rw	

Proteção – Matriz de Acesso

- Troca de domínio pode ser implementada acrescentando-se domínios à tabela:

	...	*D1*	*D2*	*D3*	*D4*
D1	...		switch		
D2	...			switch	switch
D3	...				
D4	...		switch		

Proteção – Matriz de Acesso

- Permissões podem ter bits adicionais:
  - **Cópia:** permite ao domínio **copiar** sua permissão para o objeto em outros domínios
  - **Transferência:** permite ao domínio **dar** sua permissão
  - **Cópia limitada:** permite ao domínio copiar sua transferência, mas o domínio destino não terá permissão de cópia
  - **Dono:** permite ao domínio acrescentar e remover permissões para o objeto em outros domínios

DeCSS

Proteção – Matriz de Acesso

- Implementação pode ser:
  - Tabela global:
    - simples, difícil de procurar, matriz vazia
  - Lista de acesso por objeto
  - Lista de permissões por domínio

Proteção – Remoção de Permissões

- Questões sobre remoção de permissões:
  - Imediata ou atrasada? Como saber quando a permissão foi retirada?
  - Seletiva ou geral? Remoção afeta todos usuários?
  - Parcial ou total? Todos os tipos de permissão são removidos ao mesmo tempo?
  - Temporária ou permanente?
- Remoção:
  - tabela global, lista de acessos: simples
  - lista de permissões por domínio: difícil, implica em busca em todos os domínios

Proteção – Questões Principais de Proteção

- Identificação dos domínios:
  - por usuário? anéis? processos?
- Permissões serão estáticas ou dinâmicas?
  - dinâmicas permitem proteção melhor, mas são mais complexas.
- Permissões tem que ser “on a need to know basis”
  - como dar e tirar proteções de forma segura e eficiente?
- Trocas de domínios:
  - importante, mas compromete segurança.
- Implementação:
  - tabelas globais são simples e ineficientes.
  - outros métodos são mais eficientes, mas têm problemas com remoção.