SOLUTIONS TO THE SECOND 6360 QUIZ

Jehan-François Pâris
Fall 2016

First question

- What is ***inheritance*** in Mach?

Answer

- What is ***inheritance*** in Mach?
  - Inheritance defines what happens to a range of addresses when a process forks a child.

First question

- At which level is it defined?

Answer

- At which level is it defined?
  - It is defined at the level of ***address ranges,*** that is, ranges of addresses that are all mapped by a given memory object.

First question

- Explain how Mach uses it to support both ***regular*** and ***lightweight*** processes.

Answer

- Explain how Mach uses it to support both ***regular*** and ***lightweight*** processes.
  - Mach specifies the inheritance attribute of its data segment to be
    - ***COPY*** for regular UNIX processes
    - ***SHARED*** for lightweight processes.

Second question

- Consider an ARC cache with a total capacity of 1024 pages and assume that size(T1) = target_T1 = 200 pages.
- How these two parameters would be affected if:
  - A page already present in T1 is referenced a second time?
    - **New size(T1) = ___**
    - **New target_T1 = ___**

Answer

- Consider an ARC cache with a total capacity of 1024 pages and assume that size(T1) = target_T1 = 200 pages.
- How these two parameters would be affected if:
  - A page already present in T1 is referenced a second time?
    - **New size(T1) = <u>199</u>**
    - **New target_T1 = <u>200</u>**

Second question

- Consider an ARC cache with a total capacity of 1024 pages and assume that size(T1) = target_T1 = 200 pages.
- How these two parameters would be affected if:
- A page that has never been accessed before is brought into the cache?
  - **New size(T1) = ____**
  - **New target_T1 = ____**

Answer

- Consider an ARC cache with a total capacity of 1024 pages and assume that size(T1) = target_T1 = 200 pages.
- How these two parameters would be affected if:
  - A page that has never been accessed before is brought into the cache?
    - **New size(T1) = <u>200</u>**
    - **New target_T1 = <u>200</u>**

Third question

- What ***must happen*** before Proof Carrying Code becomes widely used?

Answer

- What ***must happen*** before Proof Carrying Code becomes widely used?
  - We must find a ***cost-effective way*** to construct ***safety proofs*** for non-trivial extensions.

Fourth question

- What problem do Corey ***kernel cores*** address?

Answer

- What problem do Corey ***kernel cores*** address?
  - In most OSes, system calls are executed on the core of the invoking process
    - Bad idea if the system call needs to access large shared data structures

Fourth question

- How do they solve that problem?

Answer

- How do they solve that problem?
  - Kernel cores let applications dedicate cores to run specific kernel functions
    - Avoids inter-core contention over the data these functions access

Fifth question

- What are the two ways a malicious extension could defeat Nooks?

Answer

- What are the two ways a malicious extension could defeat Nooks?
  - A malicious extension could switch back to the kernel's page table, which would give full access to the whole kernel address space
  - It could also misuse DMA

Sixth question

- Consider a virtual memory system with
  - 4 KB pages
  - 8 GB of RAM
  - A TLB with 512 entries.
- What would be the ***TLB coverage*** of this architecture?

Answer

- Consider a virtual memory system with
  - 4 KB pages
  - 8 GB of RAM
  - A TLB with 512 entries.
- What would be the ***TLB coverage*** of this architecture?

  - **512×4KB = 2 MB**

Sixth question

- Consider a virtual memory system with
  - 4 KB pages
  - 8 GB of RAM
  - A TLB with 512 entries.
- Assume now we add to this TLB ***eight*** additional entries that can only map 1MB superpages, what would be the coverage of the new TLB***?***

Answer

- Consider a virtual memory system with
  - 4 KB pages
  - 8 GB of RAM
  - A TLB with 512 entries.
- Assume now we add to this TLB ***eight*** additional entries that can only map 1MB superpages, what would be the coverage of the new TLB***?***
  - **512×4KB + 8×1MB= 10 MB**