SOLUTIONS TO THE FIRST 6360 QUIZ

Jehan-François Pâris
Fall 2016
First question

- Why is the UNIX fork() system call so expensive?
Answer

- Why is the UNIX fork() system call **so expensive**
  - Because it has to
    - Create a new address space
    - Populate it with all the pages that were present in the parent’s address space
Second question

- How many lines will the following program print?

```c
#include <stdio.h>
#include <unistd.h>
void main() {
    fork(); fork(); fork();
    printf("Done!\n");
}
```
Answer

- How many lines will the following program print?

```
#include <stdio.h>
#include <unistd.h>
void main() {
  fork(); fork();
  printf("Done!\n");
}
```

- Four lines
Alternate second question

- How many lines will the following program print?

```c
#include <stdio.h>
#include <unistd.h>
void main() {
    fork();
    printf("Done!\n");
}
```
Answer

- How many lines will the following program print?

```c
#include <stdio.h>
#include <unistd.h>
void main() {
    fork();
    printf("Done!\n");
}
```

- Two lines
Third question

- What is the purpose of the UNIX *set-userid bit*?
Answer

- What is the purpose of the UNIX `set-userid bit`?

  - It specifies that a given program must be executed
    - With the access rights of its owner
    - Instead of the access rights of who started the program
Fourth question

- What is the purpose of *block fragments* in the Unix Fast File System?
Answer

- What is the purpose of *block fragments* in the Unix Fast File System?
  - To reduce internal fragmentation
  - Allows the file system to allocate half block and quarter blocks to small files and the tail ends of larger files.
Fifth question

- Where does Linux store its file access control lists?
Answer

- Where does Linux store its *file access control lists*?

  - In the i-node of each file
Sixth question

- A 32-bit FFS file system has a block size of 4 kilobytes.
- What is the size of the largest file that can be accessed:
  - Directly from the file i-node?
  - With at most one level of indirection?
Answer

- A 32-bit FFS file system has a block size of 4 kilobytes.

- What is the size of the largest file that can be accessed:
  - Directly from the file i-node?
    - First 12 blocks of the file
    - $12 \times 4\text{KB} = 48\text{KB}$
A 32-bit FFS file system has a block size of 4 kilobytes.

What is the size of the largest file that can be accessed:

- With at most one level of indirection?
  - Must add \((4\text{KB}/4)\times4\text{KB} = 4 \text{ MB}\)
  - Total is 4MB + 48 KB
A 32-bit FFS file system has a block size of 8 kilobytes.

What is the size of the largest file that can be accessed:

- Directly from the file i-node?
- With at most one level of indirection?
Answer

- A 32-bit FFS file system has a block size of 8 kilobytes.

- What is the size of the largest file that can be accessed:
  - Directly from the file i-node?
    - First 12 blocks of the file
    - 12×8KB = 96KB
Answer

- A 32-bit FFS file system has a block size of 4 kilobytes.

- What is the size of the largest file that can be accessed:
  - With at most one level of indirection?
    - Must add \( \frac{8\text{KB}}{4} \times 8\text{KB} = 16 \text{ MB} \)
    - Total is 16MB + 96 KB
Seventh question

- What is the main disadvantage of using the \textit{page valid bit} to simulate a missing \textit{page reference bit}?
Answer

- What is the main disadvantage of using the *page valid bit* to simulate a missing *page reference bit*?
  - Setting the simulated page-referenced bit back to one requires kernel intervention
    - *Two context switches*
Seventh question

- Consider the *two-handed* BSD clock replacement policy.

  - What happens when the *first hand* of the clock reaches a *valid page*?
Seventh question

Consider the two-handed BSD clock replacement policy.

What happens when the first hand of the clock reaches a valid page?
Answer

- Consider the *two-handed BSD clock replacement policy*.

  - What happens when the *first hand* of the clock reaches a *valid page*?

    - *It marks it invalid*
Answer

- Consider the *two-handed BSD clock replacement policy*.

- What happens when the *second hand* of the clock reaches a *valid page*?
Eighth question

- Consider the two-handed BSD clock replacement policy.

- What happens when the first hand of the clock reaches a valid page?

  - It ignores it.
Answer

Consider the *two-handed BSD clock replacement policy*.

- What happens when the *second hand* of the clock reaches a page that was *marked invalid*?
Answer

Consider the *two-handed BSD clock replacement policy*. 

- What happens when the *second hand* of the clock reaches a page that was *marked invalid*?

  - *It expels it.*