CS323 Operating Systems

Midterm

Mathias Payer and Sanidhya Kashyap

EPFL, Fall 2021

Scheduling (1a, 1b, 1c)

- Pros/cons of implementing scheduling in user land?
  - Pros: lightweight, flexible policies
  - Cons: No preemption support, must reimplement synchronization, imprecise time measurements
- Context switch: why cheaper between threads than processes?
  - No need to change address space
- Context switch: why is thread switching not free?
  - CPU state/thread context must still be swapped
- Why do we need an idle process?
  - Simplifies scheduling, always ready to run

Scheduling (1d)

```
int a = 0;
__thread int b = 1;
void doit() {
  pid_t pid = spawn();
  if (pid != 0) {
    wait(pid);
    a = 2;
    b = 3;
  } else {
    a = 4;
    b = 5;
  }
  printf("a: %d, b: %d\n", a, b);
  exit();
}
```

- Code: thread: 4, 5; 2, 3; process: 4, 5; 2, 3

Scheduling (2)

Tasks: (0, 2, 1), (0, 4, 2), (1, 3, 3), (4, 1, 4), (4, 3, 5), (5, 1, 6), (6, 2, 7).

- First In First Out: 1, 2, 3, 4, 5, 6, 7
- Shortest Job First: 1, 3, 4, 6, 7, 5, 2
- Round Robin: 1, 2, 1, 3, 2, 3, 4 ,5, 2, 6, 3, 7, 5, 2, 7, 5

Virtual memory (3a)

- Fragmentation
  Definition: free memory that cannot be used. External fragmentation: visible to allocator (OS), e.g., between segments. Internal fragmentation: visible to requester, e.g., rounding if segment size is a power of 2

Virtual memory (3b, 3c)

- Assume 12 bit pages, 32 bit virtual and 32 bit physical address space, 4 byte page table entries.
- What is the size of a page table?
  - Page table size: entries * size of entry
  - entries: $2^{log(virtualaddressspace)-log(pagesize)}$
  - entries: $2^{32-12} = 2^{20} = 1MiB$
  - Page table size: $1MiB * 4B = 4MiB$
- Offset: 6b, physical/virtual page number: 10b
- First level: 5b, second level: 5b, offset: 6b

Virtual memory (3d)

Assume that virtual addresses (not page numbers) 0x0000 – 0x02FF and 0x1FC0 – 0x2FFF are allocated. How many pages of memory are required for a single, flat page table? How many are needed for a two-level page table? How many pages are required to hold the data?

```
0000.0|000.00|00.0000
0000.0|010.11|11.1111 0x02FF
0001.1|111.11|00.0000 0x1FC0
0010.1|111.11|11.1111 0x2FFF
```

- Flat: 2^10 (#entries)/2^5 (entries per page) = 2^5 pages
- 2 layer: 1 top, 4 2nd = 5 pages
- Data: 12 + 1 + 32 + 32 = 77 pages

Concurrent programming (4a)

- 1-1 copy of the slides

Concurrent programming (4b)

- Dining philosophers
- `T1: lock(l1)`, `T2: lock(l2)`, `T3: lock(l3)`, `T4: lock(l4)`
- Solution: lock in increasing order (e.g., T4 locks l1 then l4)

Lab1: memory allocator

Use the provided test cases to double check, ping us for questions

Moodle quizzes:

- Each completed quiz will give you 1% credit for up to 5% of assignment credit.