CS 1550

Week 8 - Project 2 Q/A

TA: Jinpeng Zhou jiz150@pitt.edu

Project2 New Deadline

- Due: Oct 23, 11:59pm
- Late: Oct 25, 11:59pm

Using provided kernel

- Kernel files are on thoth: /u/OSLab/original
  - **System.map**
  - **bzImage**
  - **linux/unistd.h**
  - vmlinux

- Download System.map and bzImage to vm, cp them to /boot, run lilo:
  - You only need to do this once, and make sure to reboot your vm with “devel”

Building museumsim

- The museumsim.c should include the unistd.h which defines the new syscall macros __NR_cs1550_XXX

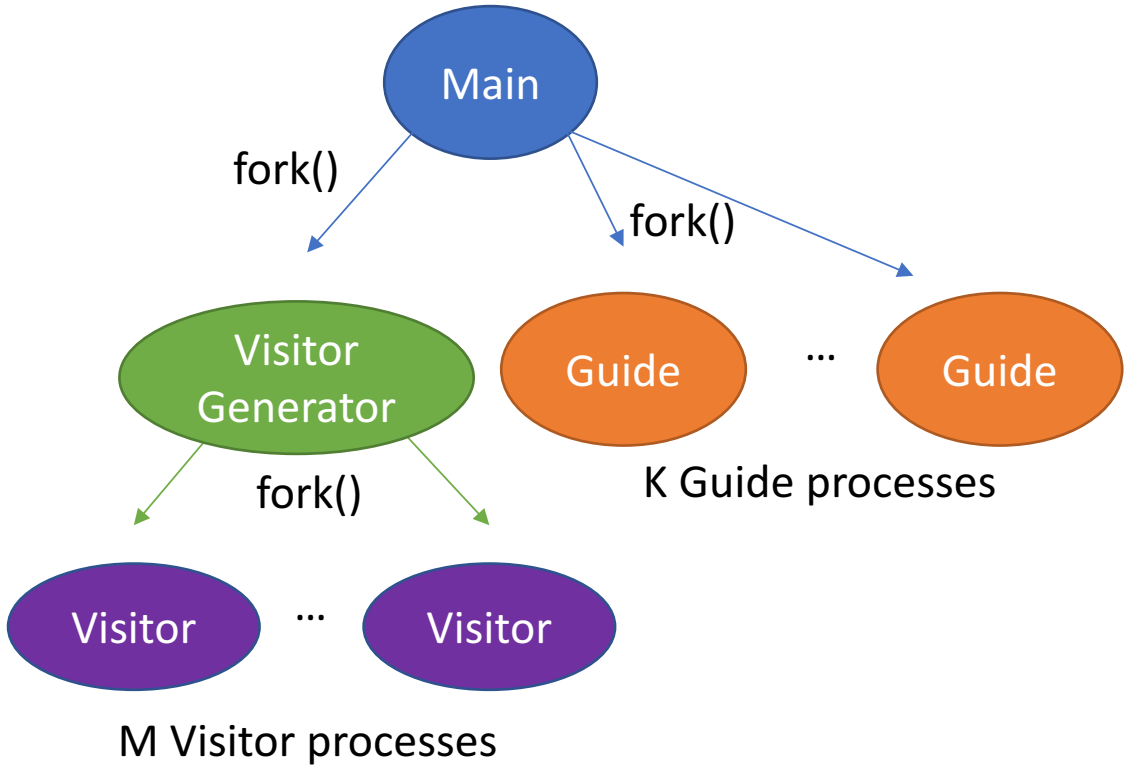
```
#include <linux/unistd.h>

int main () {

}
```

- We need to tell the compiler where it can find the linux/unistd.h. So, we'll use the following gcc command to build your program on thoth:

```
gcc -g -m32 -o museumsim -I /u/OSLab/original museumsim.c
```

Compiler will find linux/unistd.h inside /u/OSLab/original

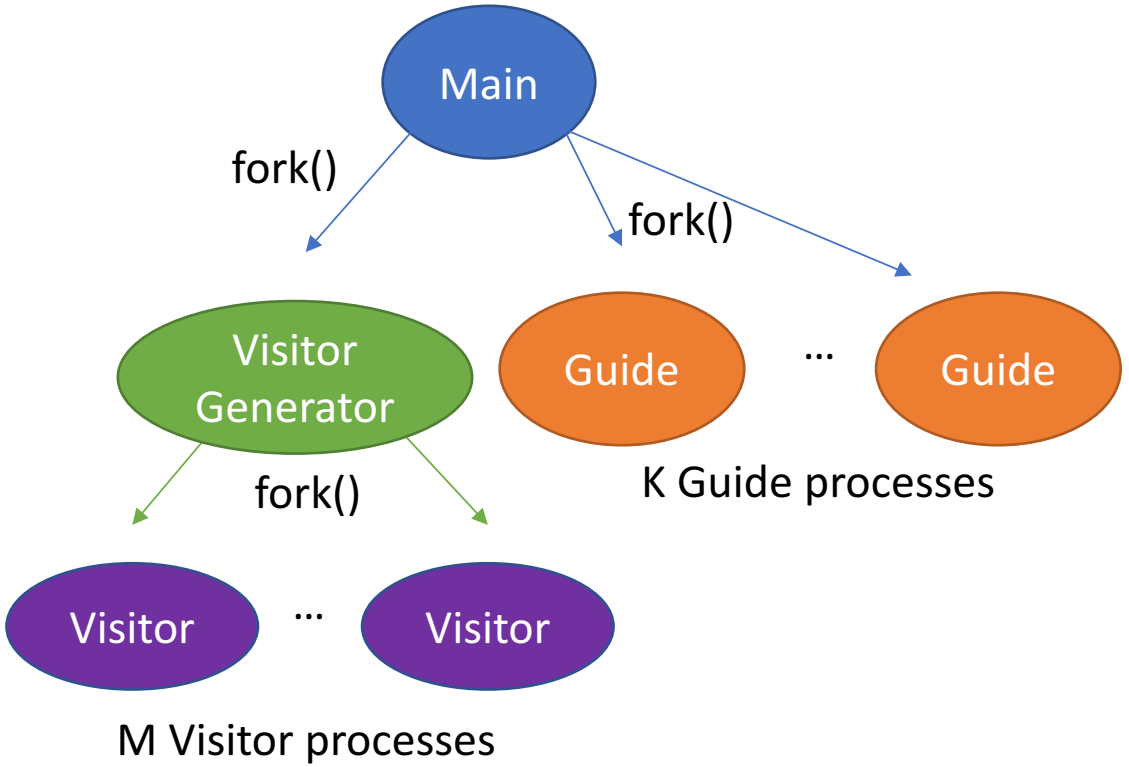
Creating visitors and guides



Managing processes

- A parent process must call “wait()” to wait for the child’s termination, such that the resource occupied by the child can be released
- In our example:
  - Main is the parent of visitor generator
    - Main should call one wait() for visitor generator
  - Main is the parent of all guides
    - Main should call K wait() for K guides
  - Visitor generator is the parent of all visitors
    - Visitor generator should call M wait() for M visitors

Make sure a parent with X children calls wait() X times.



Main process: // herein we ignore cases such as failed fork for simplicity

```
/* Main process */
pid = fork();          // Create visitor generator
if pid is 0:
        /* Visitor generator process */
        loop with M iterations:          // Create M visitors
                pid = fork();
                if pid is 0:
                        /* Visitor process */
                        Visitor arrives, tours, leaves
                        Visitor process exits
                else:
                        /* Visitor Generator process */
                        Probabilistic delay before generating next visitor
        After Visitor Generator finishes all creation, it calls M wait() for M visitors

        Visitor Generator process exists
else if pid > 0:
        /* Main process  (reused as Guide generator process) */
        Similar logic as visitor generator
Main process calls one wait for visitor generator

return 0;
```

Sharing memory among processes

```
struct shared_mem {
        int data;
};

struct shared_mem * shared = NULL;

Main() {
        // Allocation
        shared = (struct shared_mem*) mmap (NULL, sizeof(struct shared_mem), PROT_READ|PROT_WRITE, MAP_SHARED|MAP_ANONYMOUS, 0, 0);

        // Initialization
        shared->data = 1;

        …

        if (pid == 0){ // child process
                …
                shared->data = 2;
                …
        } else { // parent process
                …
                shared->data = 3;
                …
        }
}
```

Using semaphores

```c
#include <linux/unistd.h>

long create(int value, char name[32], char key[32]) {
  return syscall(__NR_cs1550_create, value, name, key);
}

long open(char name[32], char key[32]) {
  return syscall(__NR_cs1550_open, name, key);
}

long down(long sem_id) {
  return syscall(__NR_cs1550_down, sem_id);
}

long up(long sem_id) {
  return syscall(__NR_cs1550_up, sem_id);
}

long close(long sem_id) {
  return syscall(__NR_cs1550_close, sem_id);
}
```

To achieve mutual exclusive execution:

long sem_mutex = create (1, name, key);

down (sem_mutex);
// access shared data
up (sem_mutex);

To represent a type of resource:

long sem_res = create (N, name, key); // N available initially

Please refer to trafficsim.c and trafficsim-mutex.c for more examples