OSTEP Chapter 2

ECE 3600, Fall 2022

Table of Contents

1. Introduction

3 parts (pieces):

1. Virtualization: CPU, memory
2. Concurrency: threads [and processes]
3. Persistence: file system

OS Design Goals:

- Abstractions (APIs)
- Performance (minimize overhead)
- Protection (isolation)
- Reliability
- Other: security, energy-efficiency, mobility (dynamic networking)

How:

- Software: operating system (kernel)
- Hardware: user mode vs. kernel mode, privileged instructions, system calls, traps

2. cpu.c example

```
$ make
gcc -o cpu cpu.c -Wall
gcc -o mem mem.c -Wall
gcc -o threads threads.c -Wall -pthread
gcc -o io io.c -Wall
$ cat -n ostep/code/intro/cpu.c
     1  #include <stdio.h>
     2  #include <stdlib.h>
     3  #include "common.h"
     4
     5  int main(int argc, char *argv[])
     6  {
     7      if (argc != 2) {
     8          fprintf(stderr, "usage: cpu <string>\n");
     9          exit(1);
    10      }
    11      char *str = argv[1];
    12
    13      while (1) {
    14          printf("%s\n", str);
    15          Spin(1);
    16      }
    17      return 0;
    18  }
$ ./cpu abc
abc
abc
abc
^C
$
```

```
$ ./cpu A & ./cpu B & ./cpu C & ./cpu D &
[1] 3153
[2] 3154
[3] 3155
[4] 3156
$A
B
D
C
A
B
D
C
A
B
D
C
...
kill %1 %2 %3 %4
[1]   Terminated              ./cpu A
[2]   Terminated              ./cpu B
[3]-  Terminated              ./cpu C
[4]+  Terminated              ./cpu D
$
```

--> sleep() instead of Spin()

3. mem.c example

```
$ cat -n ostep/code/intro/mem.c
     1  #include <unistd.h>
     2  #include <stdio.h>
     3  #include <stdlib.h>
     4  #include "common.h"
     5
     6  int main(int argc, char *argv[]) {
     7      if (argc != 2) {
     8          fprintf(stderr, "usage: mem <value>\n");
     9          exit(1);
    10      }
    11      int *p;
    12      p = malloc(sizeof(int));
    13      assert(p != NULL);
    14      printf("(%d) addr pointed to by p: %p\n", (int) getpid(), p);
    15      *p = atoi(argv[1]); // assign value to addr stored in p
    16      while (1) {
    17          Spin(1);
    18          *p = *p + 1;
    19          printf("(%d) value of p: %d\n", getpid(), *p);
    20      }
    21      return 0;
    22  }
$ ./mem 10
(3629) addr pointed to by p: 0x55917a0fb260
(3629) value of p: 11
(3629) value of p: 12
(3629) value of p: 13
^C
$
```

```
$ ./mem 10 & ./mem 100 &
[1] 3645
[2] 3646
$ (3645) addr pointed to by p: 0x55ae4b29a260
(3646) addr pointed to by p: 0x55a15c730260
(3645) value of p: 11
(3646) value of p: 101
(3645) value of p: 12
(3646) value of p: 102
...
$ setarch $(uname --machine) --addr-no-randomize /bin
$ ./mem 10 & ./mem 100 &
[1] 3680
[2] 3681
$ (3680) addr pointed to by p: 0x555555756260
(3681) addr pointed to by p: 0x555555756260
(3680) value of p: 11
(3681) value of p: 101
(3680) value of p: 12
(3681) value of p: 102
...
```

4. threads.c example

```
$ cat -n ostep/code/intro/threads.c
     1  #include <stdio.h>
     2  #include <stdlib.h>
     3  #include "common.h"
     4  #include "common_threads.h"
     5
     6  volatile int counter = 0;
     7  int loops;
     8
     9  void *worker(void *arg) {
    10      int i;
    11      for (i = 0; i < loops; i++) {
    12          counter++;
    13      }
    14      return NULL;
    15  }
    16
    17  int main(int argc, char *argv[]) {
    18      if (argc != 2) {
    19          fprintf(stderr, "usage: threads <loops>\n");
    20          exit(1);
    21      }
    22      loops = atoi(argv[1]);
    23      pthread_t p1, p2;
    24      printf("Initial value : %d\n", counter);
    25      Pthread_create(&p1, NULL, worker, NULL);
    26      Pthread_create(&p2, NULL, worker, NULL);
    27      Pthread_join(p1, NULL);
    28      Pthread_join(p2, NULL);
    29      printf("Final value   : %d\n", counter);
    30      return 0;
    31  }
$
```

```
$ ./threads 10
Initial value : 0
Final value   : 20
$ ./threads 100
Initial value : 0
Final value   : 200
$ ./threads 1000
Initial value : 0
Final value   : 2000
$ ./threads 10000
Initial value : 0
Final value   : 19897
$ ./threads 10000
Initial value : 0
Final value   : 15562
$ ./threads 10000
Initial value : 0
Final value   : 14179
$ ./threads 100000
Initial value : 0
Final value   : 103750
```

5. io.c example

```
$ cat -n ostep/code/intro/io.c
     1  #include <stdio.h>
     2  #include <unistd.h>
     3  #include <assert.h>
     4  #include <fcntl.h>
     5  #include <sys/stat.h>
     6  #include <sys/types.h>
     7  #include <string.h>
     8
     9  int main(int argc, char *argv[]) {
    10      int fd = open("/tmp/file", O_WRONLY | O_CREAT | O_TRUNC, S_IRUSR | S_IWUSR);
    11      assert(fd >= 0);
    12      char buffer[20];
    13      sprintf(buffer, "hello world\n");
    14      int rc = write(fd, buffer, strlen(buffer));
    15      assert(rc == (strlen(buffer)));
    16      fsync(fd);
    17      close(fd);
    18      return 0;
    19  }
$ ./io
$ cat /tmp/file
hello world
$ ls -ld /tmp
drwxrwxrwt 19 root root 12288 Jun  5 11:43 /tmp
$ ls -l /tmp/file
-rw------- 1 perry perry 12 Jun  5 11:43 /tmp/file
$ ./io
$ chmod 400 /tmp/file
$ ls -l /tmp/file
-r-------- 1 perry perry 12 Jun  5 11:44 /tmp/file
$ ./io
io: io.c:11: main: Assertion `fd >= 0' failed.
Abort (core dumped)
$
```

--> perror() instead of assert()