

# CS302 OS Week4 Assignment - Report

---

Name: 刘仁杰

SID: 11911808

## 1. Read Chapter 2 of "Three Easy Pieces"

---

1. "three easy pieces" of operating systems
  - Virtualization: transform a physical resource (cpu, memory, disk, ...) into a more general easy-to-use virtual form, which further allows many programs to run and share physical resources concurrently.
  - Concurrency: working on many things concurrently in the same program or by multi-threaded programs without incurring incorrect results.
  - Persistence: How the os take control of file system to store data persistently, access device, enhance performance with typical mechanisms and recover data from crash for failure.
2. How do these "three easy pieces" map to the chapters in the "dinosaur book"?
  - Virtualization: Chapter 5,10
  - Concurrency: Chapter 4,6,7,8
  - Persistence: Chapter 12,13,14,15

## 2. What happens during context switch in detail?

---

- Process A is running
- Interrupted by timer
- Hardware saves A's registers onto its kernel stack and switch to the kernel mode
- Trap handler call `switch()` routine which saves current register values into the process structure of A and restores the registers of Process B from its process structure entry.
- Change the stack pointer to use B's kernel stack
- OS returns-from-trap, which restores B's registers from B's kernel stack and starts running Process B

## 3. Read slides "L03 Processes I" and "L04 Processes II" and answer questions:

---

(1) Explain what happens when the kernel handles the `fork()` system call

- When `fork()` system call is invoked, the program switch from user mode to kernel mode (save and restore context)
- In OS kernel, new address space is created for the child process, and copy the kernel space (PCB) of parent process to the newly created address space
- OS kernel does the kernel update of the child kernel space, which includes PID, running time, and a pointer to its parent
- OS kernel will also create a pointer in parent process's kernel space which points to the forked child
- Add the child process to the task list
- User space of the child process is also copied and updated
- `fork()` is completed, return value is set in both the parent process and newly forked child process respectively to be the child process PID and 0
- CPU scheduler decide which process to be executed next

- If parent process is to be executed next, the child PID is returned, program switch back to user mode and the process is continued
- If child process is to be executed next, there will be a context switch from parent process to child process which involves PCB reload, pc rebase and switch from kernel mode to user mode, starting running process B

(2) Explain what happens when the kernel handles the `exit()` system call

- When `exit()` system call is invoked, the program switch from user mode to kernel mode
- The kernel frees all allocated memory in kernel space, except the PID and the list of opened files are closed
- The kernel frees everything on the user-space memory about the concerned process, including program code and allocated memory
- The kernel sends a `SIGCHLD` signal to its parent process which notifies the child process has now terminated
- If its parent process invoked `wait()` system call
  - the kernel will register a signal handling routine for parent process and the process is blocked
  - When it receives `SIGCHLD`, the corresponding signal handling routine is invoked
  - The `SIGCHLD` handler accepts and removes the `SIGCHLD` signal and destroys the child process in the kernel-space (remove it from process table, task-list, etc.)
  - Now the child process is truly dead and recycled
- If the parent process hasn't invoked `wait()` system call
  - the parent process doesn't respond to the `SIGCHLD` signal
  - Thus, the child process becomes a zombie until the parent process invoked `wait()` and receive its `SIGCHLD` signal

## 4. What are the three methods of transferring the control of the CPU from a user process to OS kernel?

- system call: system calls in user program require the switch from user mode to kernel mode, which typically ask for the control of kernel resources
- interrupt: Typically invoked by interrupt signals, the signals received is passed to trap handler who further dispatches the task to the corresponding interrupt handler function
- exception: Typically invoked by exception signals, the signals received is passed to trap handler who further dispatches the task to the corresponding exception handler function

## 5. Describe the life cycle of a process

- Birth: Except the first process "init", every process is created using `fork()`
- After one process is just forked or OS scheduler choose another process to run or a process is just returning from blocked states, the process will enter `ready state`
- When OS chooses this process to be running on the CPU, the process state is adjusted to `running state`
- While the process the running, it may wait for some resources, which will cause the process to be in `blocking state`
- In `blocking state`, when response arrives, the status of the process changes back to `ready state`
- When the process goes to an end or is forced to terminate, the process enters `Zombie(terminated) state`

## 6. Realize a shell of your own in myshell.c through fork () + exec () + wait ()

- start the shell

```
lrj11911808@lrj-virtual-machine: ~/CS302_OS/assignment/week4ass
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ gcc -o myshell ./myshell.c
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ ./myshell
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$
```

- Basic Instructions -- ps, ls, ls -al, pwd

```
lrj11911808@lrj-virtual-machine: ~/CS302_OS/assignment/week4ass
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ gcc -o myshell ./myshell.c
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ ./myshell
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ ps
  PID TTY          TIME CMD
 29360 pts/0    00:00:00 bash
 33078 pts/0    00:00:00 myshell
 33196 pts/0    00:00:00 ps
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ ls
myshell  myshell.c  report.md  test  test.c
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ ls -a
.  ..  myshell  myshell.c  report.md  test  test.c
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ ls -al
total 68
drwxrwxr-x 2 lrj11911808 lrj11911808 4096 3月 23 19:45 .
drwxrwxr-x 5 lrj11911808 lrj11911808 4096 3月 23 16:59 ..
-rwxrwxr-x 1 lrj11911808 lrj11911808 18056 3月 23 19:45 myshell
-rw-rw-r-- 1 lrj11911808 lrj11911808 5181 3月 23 19:26 myshell.c
-rw-rw-r-- 1 lrj11911808 lrj11911808 5358 3月 23 16:59 report.md
-rwxrwxr-x 1 lrj11911808 lrj11911808 16736 3月 23 16:59 test
-rw-rw-r-- 1 lrj11911808 lrj11911808 154 3月 23 16:59 test.c
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ pwd
/home/lrj11911808/CS302_OS/assignment/week4ass
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$
```

- Basic Instructions -- which, echo, touch, cat, rm

```
lrj11911808@lrj-virtual-machine: ~/CS302_OS/assignment/week4ass
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ ./myshell
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ which which
/usr/bin/which
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ which ls
/usr/bin/ls
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ echo hello
hello
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ touch temp.out
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ ls
myshell  myshell.c  report.md  temp.out  test  test.c
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ echo hello > temp.out
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ cat temp.out
hello
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ rm temp.out
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ ls
myshell  myshell.c  report.md  test  test.c
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$
```

- cd .., cd, cd ~, cd /home, cd /

```
lrj11911808@lrj-virtual-machine: ~/CS302_OS/assignment/week4ass
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ ./myshell
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ cd ..
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment$ cd ../../
lrj11911808@lrj-virtual-machine:~$ cd ./CS302_OS/assignment
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment$ cd
lrj11911808@lrj-virtual-machine:~$ cd ./CS302_OS
lrj11911808@lrj-virtual-machine:~/CS302_OS$ cd ~
lrj11911808@lrj-virtual-machine:~$ cd /home
lrj11911808@lrj-virtual-machine:/home$ cd /
lrj11911808@lrj-virtual-machine:/$ cd
lrj11911808@lrj-virtual-machine:~$
```

- exit

```
lrj11911808@lrj-virtual-machine: ~/CS302_OS/assignment/week4ass
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ ./myshell
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ ls
myshell myshell.c report.md test test.c
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ ps
  PID TTY          TIME CMD
 29360 pts/0    00:00:00 bash
 33345 pts/0    00:00:00 myshell
 33347 pts/0    00:00:00 ps
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ ls -al
total 68
drwxrwxr-x 2 lrj11911808 lrj11911808 4096 3月 23 19:55 .
drwxrwxr-x 5 lrj11911808 lrj11911808 4096 3月 23 16:59 ..
-rwxrwxr-x 1 lrj11911808 lrj11911808 18056 3月 23 19:45 myshell
-rw-rw-r-- 1 lrj11911808 lrj11911808 5181 3月 23 19:26 myshell.c
-rw-rw-r-- 1 lrj11911808 lrj11911808 5358 3月 23 16:59 report.md
-rwxrwxr-x 1 lrj11911808 lrj11911808 16736 3月 23 16:59 test
-rw-rw-r-- 1 lrj11911808 lrj11911808 154 3月 23 16:59 test.c
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ exit
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$
```

- background process

One can launch background process using `&` and the terminal goes without being blocked

```
lrj11911808@lrj-virtual-machine: ~/CS302_OS/assignment/week4ass
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ ./myshell
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ ./test &
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ active

lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ active
active
active
active

lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ active
active
active

lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ active
active

lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ ./test &
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ ps
  PID TTY          TIME CMD
 29360 pts/0    00:00:00 bash
 33360 pts/0    00:00:00 myshell
 33361 pts/0    00:00:00 test <defunct>
 33364 pts/0    00:00:00 test
 33365 pts/0    00:00:00 ps
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ active
```

- pipe

```
lrj11911808@lrj-virtual-machine: ~/CS302_OS/assignment/week4ass
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ ./myshell
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ ls | grep shell
myshell
myshell.c
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ cat myshell.c | grep shell
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ cat myshell.c | grep fork
    if (fork() == 0)
        int pid = fork();
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ ls | sort
myshell
myshell.c
report.md
test
test.c
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$
```

- FIFO

one can launch a FIFO as usual and it will be blocked until someone reads

```
lrj11911808@lrj-virtual-machine: ~/CS302_OS/assignment/week4ass
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ ./myshell
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ mkfifo myfifo
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ ls
myfifo myshell myshell.c report.md test test.c
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ ls | grep fifo
myfifo
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ echo hello > myfifo
```

```
lrj11911808@lrj-virtual-machine: ~/CS302_OS/assignment/week4ass
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ ./myshell
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ mkfifo myfifo
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ ls
myfifo myshell myshell.c report.md test test.c
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ ls | grep fifo
myfifo
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ echo hello > myfifo
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$

lrj11911808@lrj-virtual-machine: ~/CS302_OS/assignment/week4ass
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ ./myshell
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ cat myfifo
hello
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$
```

- ctrl + c

ctrl + c can terminate child process while it loses efficacy on the shell itself

```
lrj11911808@lrj-virtual-machine: ~/CS302_OS/assignment/week4ass
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ ./myshell
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ ./test
active
active
active
^C
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ ^C^C^C
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ ^C^C^C
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ ^C
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$ ^C
lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week4ass$
```

