top

2023年8月10日 9:30

```
top - 10:00:10 up 30 min,  3 users,  load average: 0.13, 0.08, 0.08
Tasks: 290 total,   1 running, 289 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni, 99.8 id,  0.0 wa,  0.0 hi,  0.2 si,  0.0 st
MiB Mem :   7903.8 total,   5624.5 free,   1076.5 used,   1202.8 buff/cache
MiB Swap:   2048.0 total,   2048.0 free,      0.0 used.   6530.1 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   1028 mysql     20   0 1755120 396504  38368 S   1.0   4.9   0:20.39 mysqld
    955 redis     20   0   61428   4660   3364 S   0.3   0.1   0:02.52 redis-server
   2540 liao      20   0   20624   4000   3228 R   0.3   0.0   0:00.05 top
      1 root      20   0  168448  11624   8336 S   0.0   0.1   0:04.07 systemd
```

```
   us, user    : time running un-niced user processes
   sy, system  : time running kernel processes
   ni, nice    : time running niced user processes
   id, idle    : time spent in the kernel idle handler
   wa, IO-wait : time waiting for I/O completion
   hi : time spent servicing hardware interrupts
   si : time spent servicing software interrupts
   st : time stolen from this vm by the hypervisor
```

qemu -

优先级系统

2023年8月10日 10:16

PRI → 140级 [-40,99] 数值越大优先级别越低

时间片长度.

调度策略. { 高优先级 实时. { FIFO / RR [-40,59]
低优先级 普通 CFS [60,99]

NICE. 普通用户用. 不改变调度策略情况下. 间接修改优先级

NICE	PRI
-20	60
0	80
19	99

[-20,19]

nice renice

2023年8月10日 10:28

```
liao:LinuxDay13$ nice -n 20 ./0_homework
^C
liao:LinuxDay13$ ./0_homework
^C
liao:LinuxDay13$ nice -n -10 ./0_homework
nice: cannot set niceness: Permission denied
^C
liao:LinuxDay13$ sudo nice -n -10 ./0_homework
[sudo] password for liao:
^C
liao:LinuxDay13$ sudo nice -n -20 ./0_homework
^C
liao:LinuxDay13$ sudo nice -n -21 ./0_homework
^C
```

①启动 nice >= 0, 不用root

② nice < 0 用root

③ nice [-20,19]

renice

2023年8月10日 10:32

```
liao:~$ renice -n 10 -p 3013
3013 (process ID) old priority 0, new priority 10
liao:~$
```

增加数值 不用root

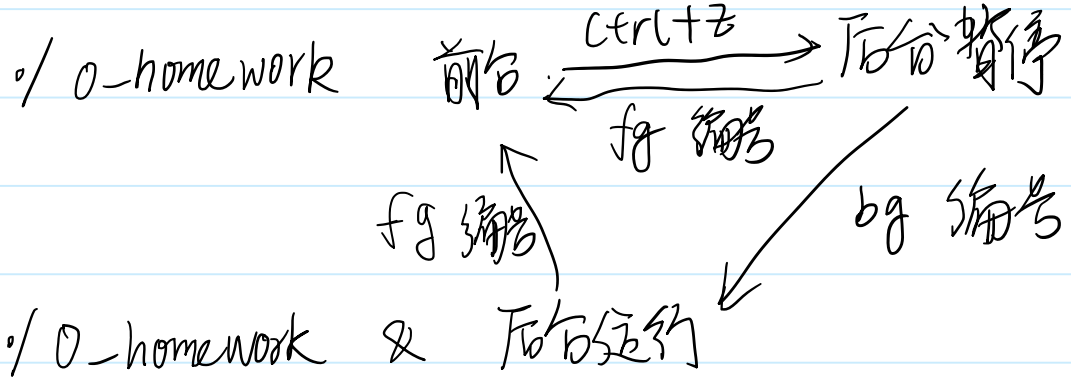
减少数值，要root

前台和后台

2023年8月10日 10:35

前台：可以响应键盘中断. Ctrl+C

后台：不可响应.



jobs 获取本窗口的前后台进程

kill -9 pid. 终止

库函数 一个可执行程序，在执行过程中创建多个进程

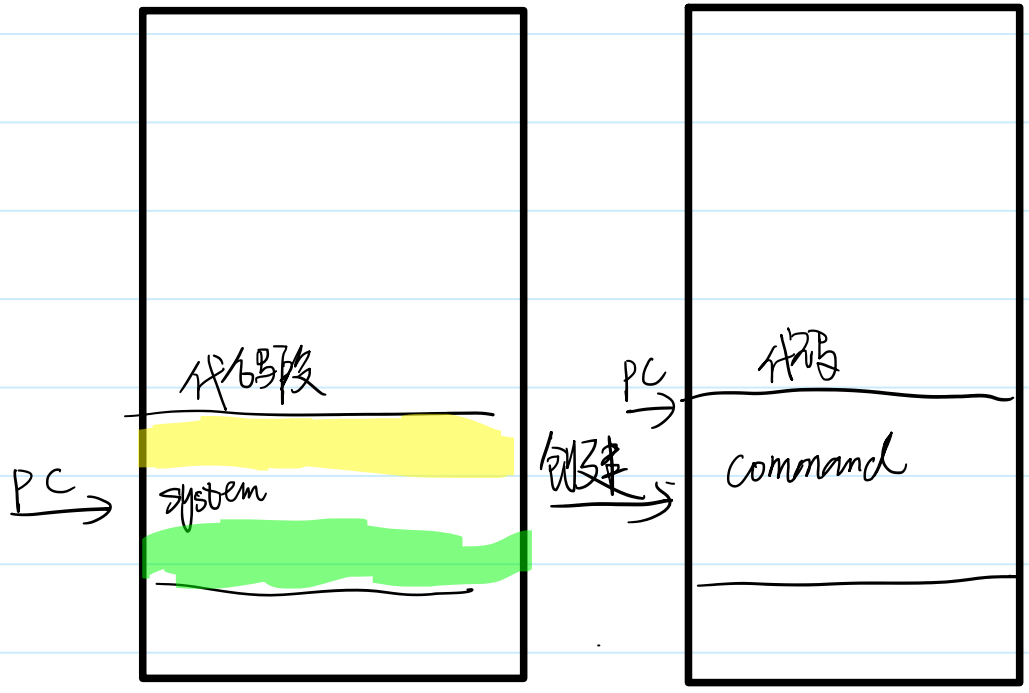
2023年8月10日 11:10

```
NAME
       system - execute a shell command

SYNOPSIS
       #include <stdlib.h>

       int system(const char *command);
```

①创建子进程 command 分配.

② 子进程执行 command

③ 父进程等待子进程完成任务.

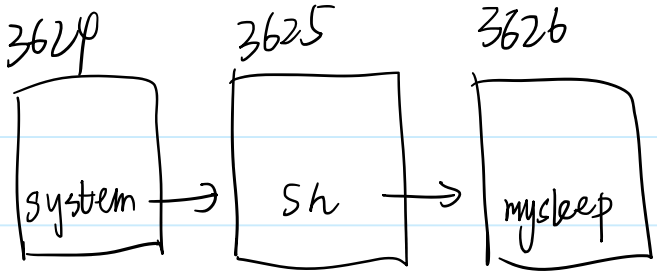



```c
int main(int argc, char *argv[])
{
    printf("before\n");
    system("ls -al");
    printf("after\n");
    return 0;
}
```

system 创建了进程

2023年8月10日 11:18

```
        3624    2528
./2_system
        3625    3624
sh -c ./mysleep
        3626    3625
./mysleep
```




```c
int main(int argc, char *argv[])
{
    printf("before\n");
    system("./mysleep");
    printf("after\n");
    return 0;
}
```

```c
int main(int argc, char *argv[])
{
    sleep(10);
    printf("sleep over!\n");
    return 0;
}
```

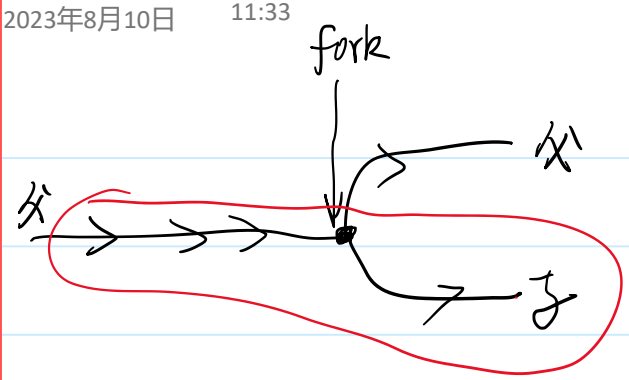
system调用其他语言的代码

2023年8月10日 11:25

```c
int main(int argc, char *argv[])
{
    printf("before\n");
    system("python3 hello.py");
    printf("after\n");
    return 0;
}
```

fork 创建子进程的系统调用

2023年8月10日 11:33



fork通过复制父进程创建子进程

NAME

fork - create a child process

SYNOPSIS

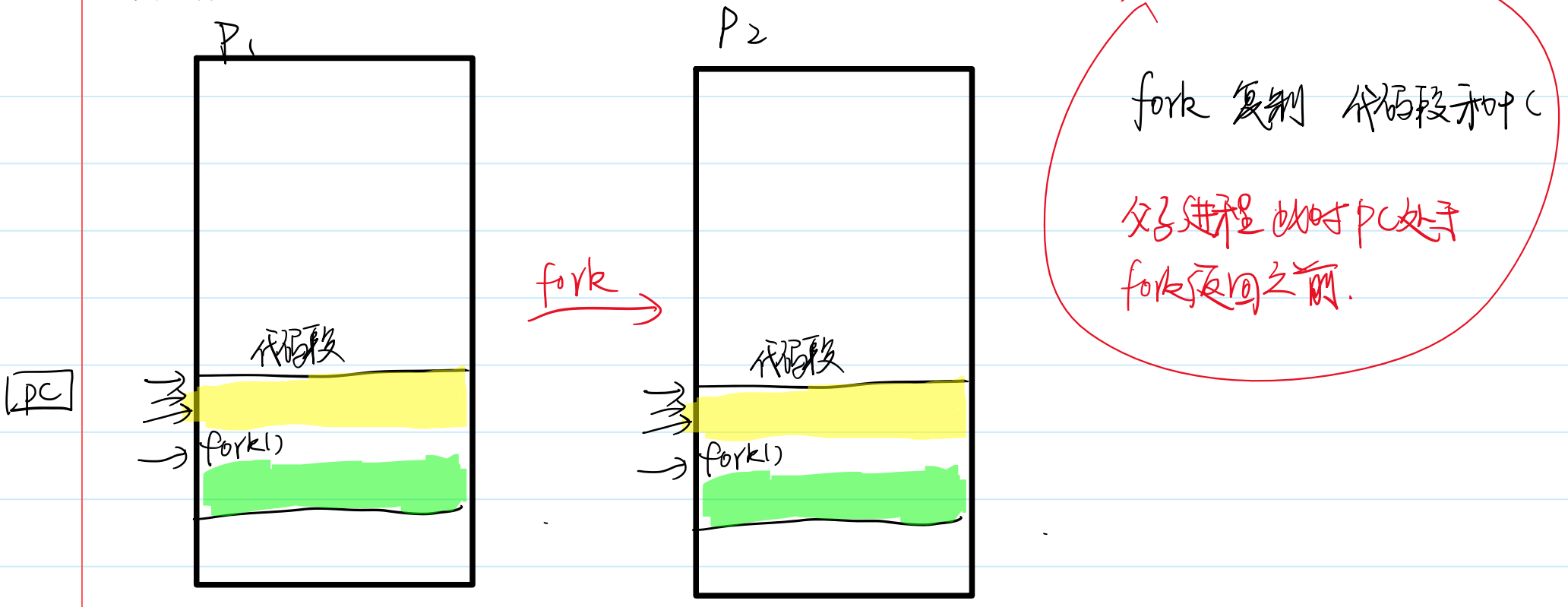
```
#include <sys/types.h>
#include <unistd.h>

pid_t fork(void);
```

DESCRIPTION

fork() creates a new process by duplicating the calling process. The new process is referred to as the child process. The calling process is referred to as the parent process.

fork的模型

2023年8月10日 11:37



fork 复制 代码段和PC

父子进程此时PC处于fork返回之前.

```c
int main(int argc, char *argv[])
{
    printf("Hello\n");
    fork();
    printf("World\n");
    return 0;
}
```

```
liao:LinuxDay13$ ./4_fork
Hello
World
World
```

fork的返回值

2023年8月10日 11:46

the PID of the child process is returned in the parent, and 0 is returned in the child.

如何让父进程和子进程做不一样的事?

① fork()的返回值

② 共享.

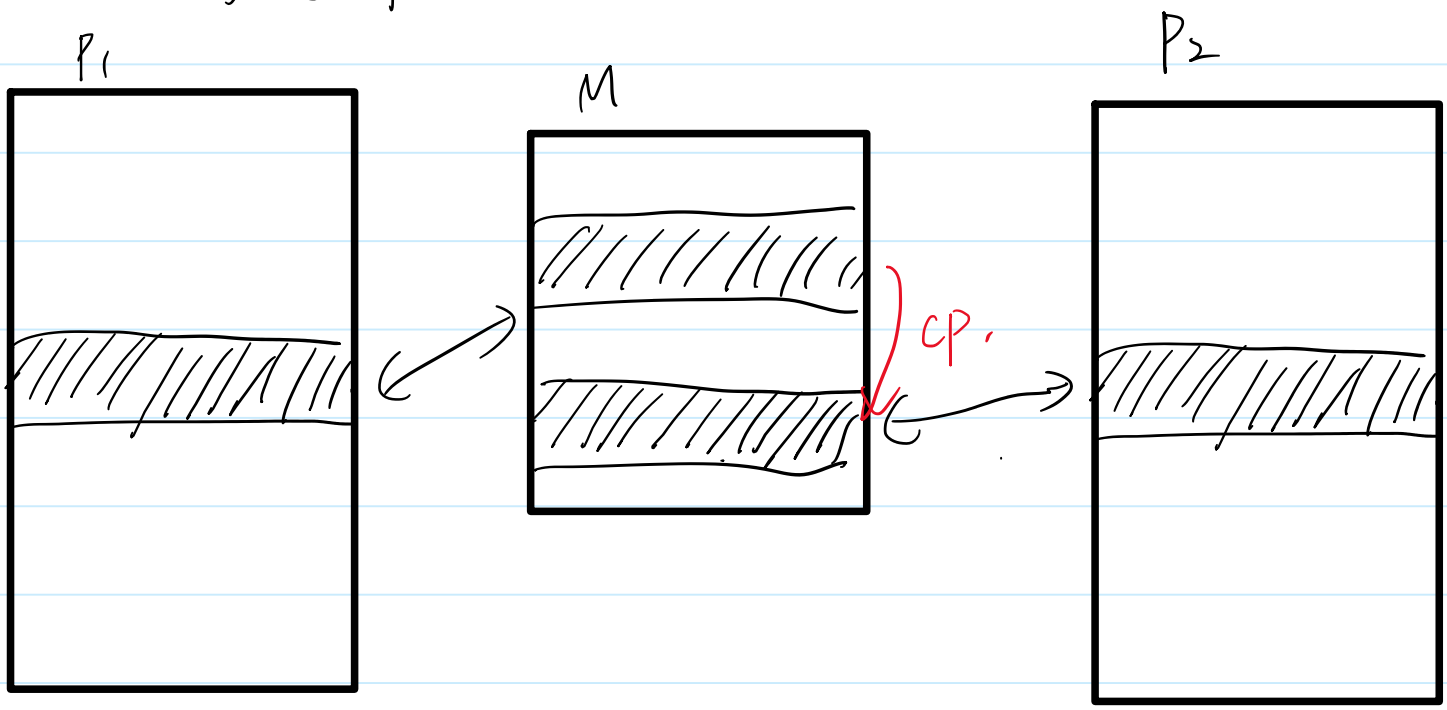
5_fork.c

```c
#include <52func.h>
int main(int argc, char *argv[])
{
    printf("Hello\n");
    pid_t pid = fork();
    if(pid == 0){
        // 子进程
        printf("World2\n");
    }
    else{
        // 父进程
        printf("World1\n");
        sleep(1);
    }
    return 0;
}
```

fork实现过程中的优化

2023年8月10日 11:57

实现过程，对于使用者无影响.



写时复制

2023年8月10日 12:03