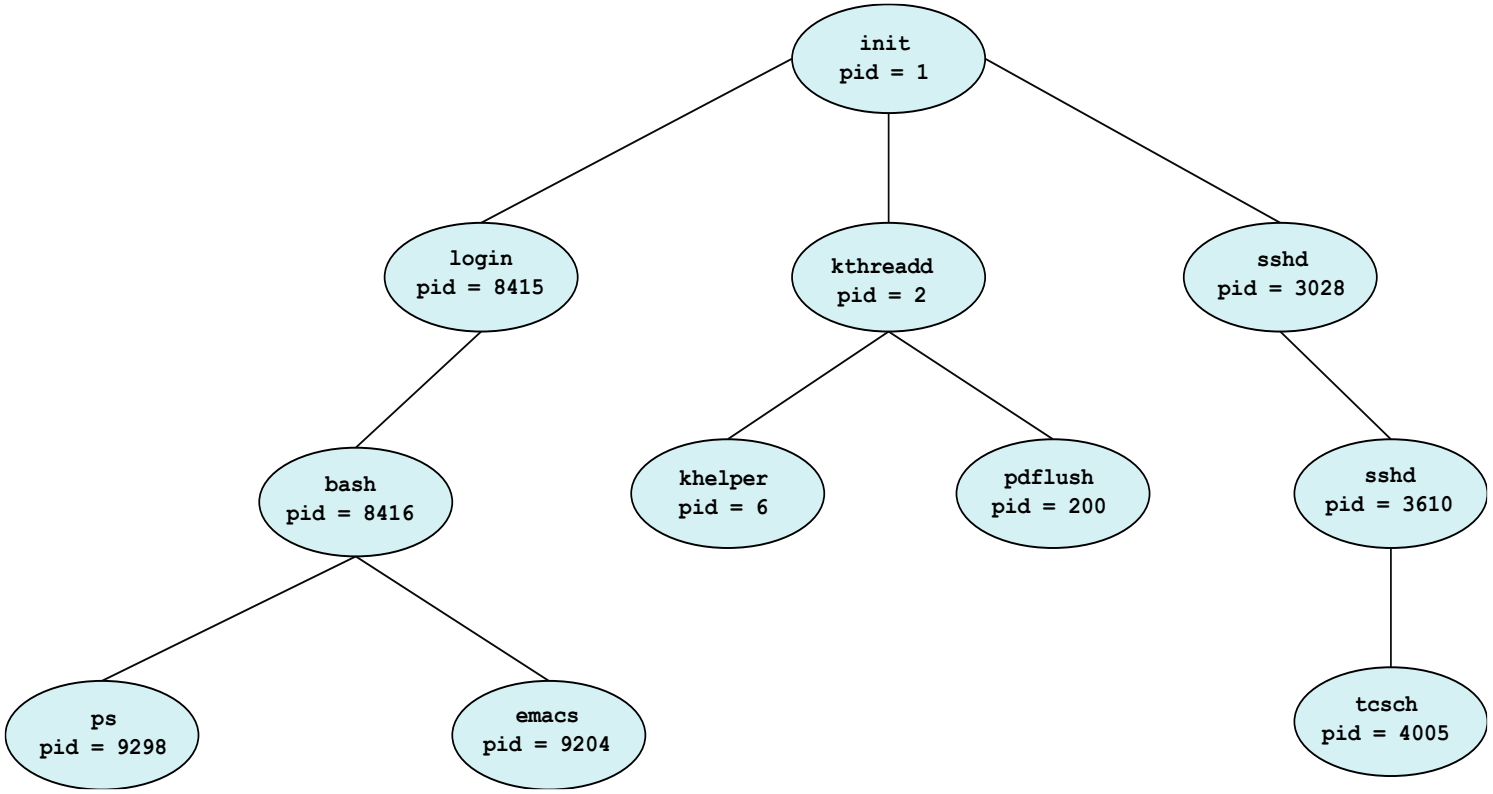
Fork

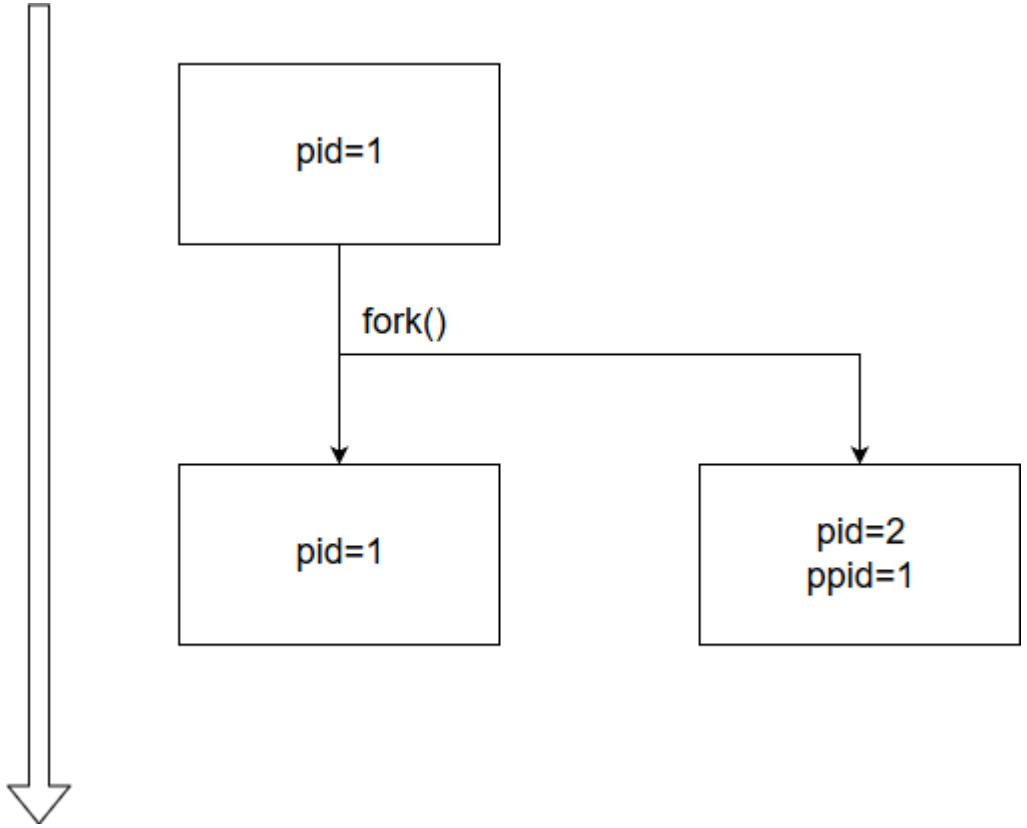
分叉；分歧

A tree of processes in Linux



- 所有进程都是init进程(pid=1)的后代进程；
- Init进程在加载操作系统时创建；
- 处init进程外，所有其他进程都是通过fork()或者类似的函数来创建

fork()：



通过拷贝当前进程创建一个子进程，子进程与父进程的区别在于PID、PPID和某些资源和统计量（比如挂起的信号）。

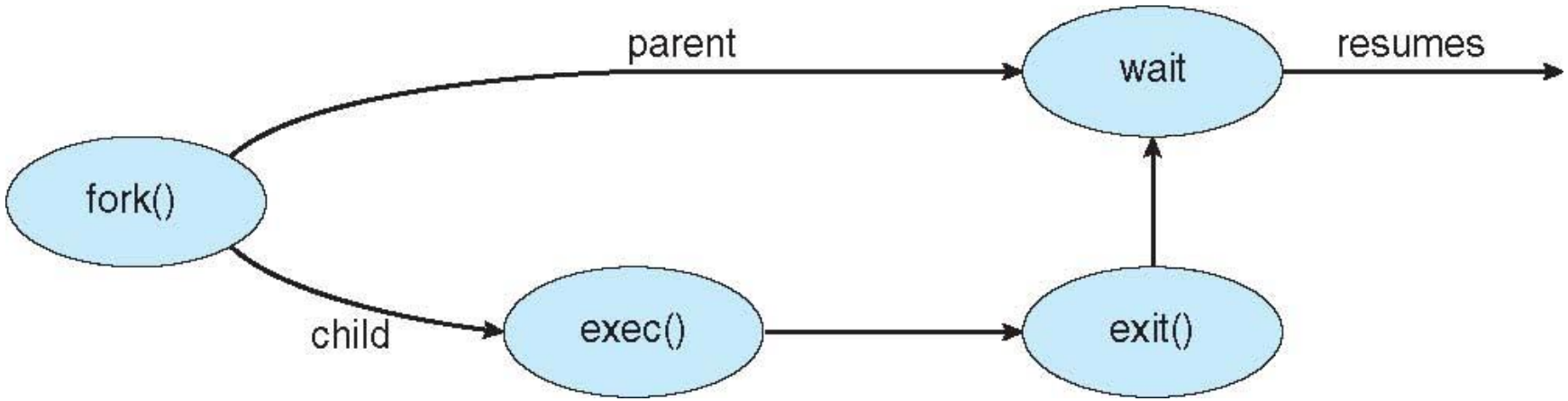
fork()工作流程:

- 为新进程创建内核栈、进程描述符task_struct，与父进程完全相同；
- 检查并确保这个子进程没有超出资源，如当前用户所能拥有的进程数，内存等；
- 子进程设置task_struct，开始与父进程区别开来；
- 更新子进程的某些flags，比如PF_FORKNOEXEC
- 拷贝或共享打开的文件、文件系统信息、信号处理函数、进程地址空间和命名空间等；
- 唤醒并运行子进程

Copy-on-write

- 子进程不一定需要父进程的数据；
- 推迟甚至免除拷贝数据，子进程以只读方式共享父进程的资源；
- 在需要写入的时候，数据才会被复制；
- fork()实际开销小（赋值父进程的页表，给子进程创建唯一task_struct）

- 先运行子进程（避免拷贝）
- 实际应用中，由系统调度来控制
- 多核环境下有可能两进程同时运行



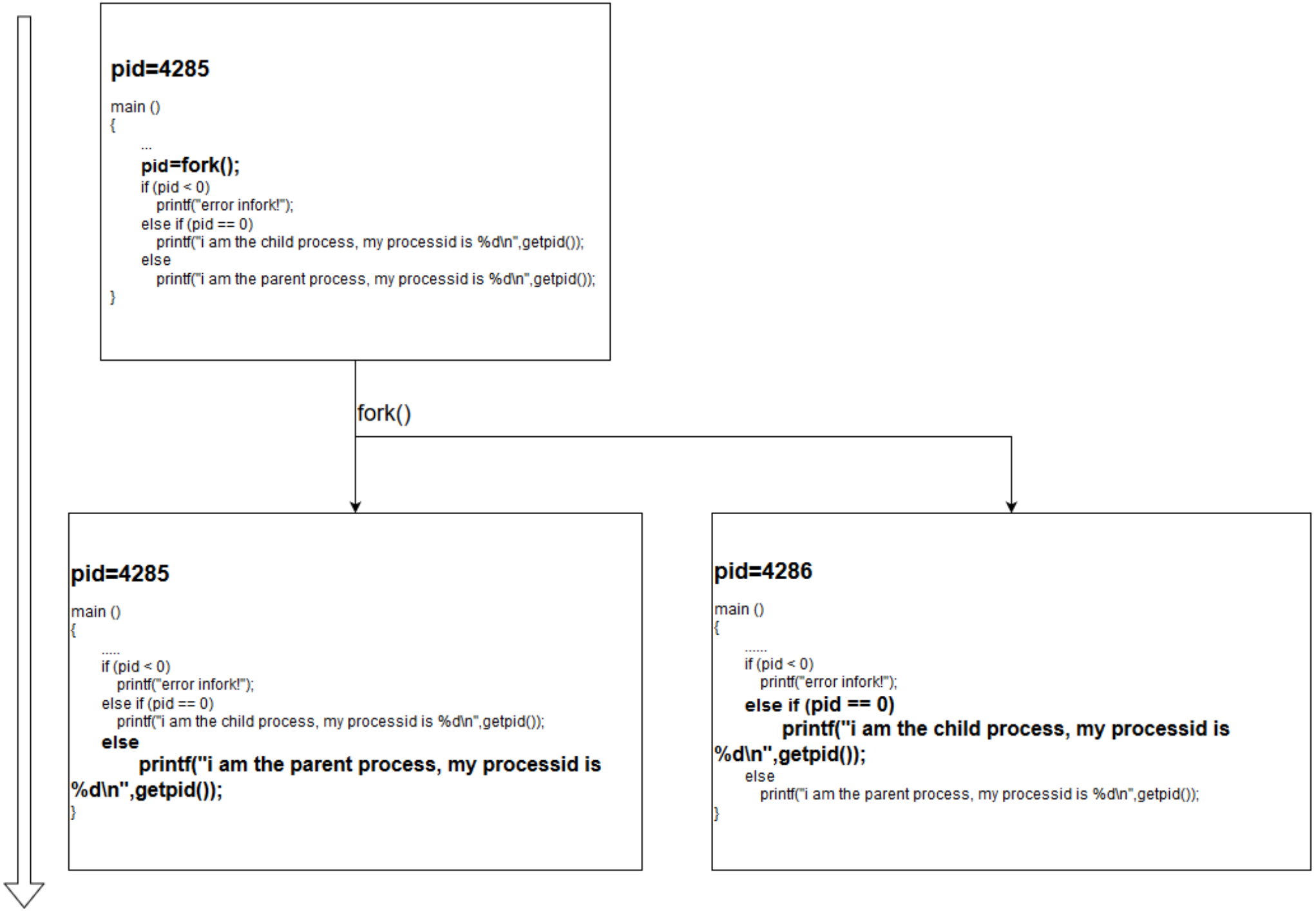
fork() 返回值

- >0：　父进程；值等于子进程的pid；
- ==0：子进程；
- <0：　fork()失败

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

main ()
{
        pid_t pid;
        pid=fork();

        if (pid < 0)
                printf("error in fork!");
        else if (pid == 0)
                printf("i am the child process, my process id is %d\n",getpid());
        else
                printf("i am the parent process, my process id is %d\n",getpid());
}
```

输出结果：

```
i am the child process, my process id is 4286

i am the parent process, my process id is 4285
```


pid=4285
main ()
{
...
pid=fork();
if (pid < 0)
printf("error infork!");
else if (pid == 0)
printf("i am the child process, my processid is %d\n",getpid());
else
printf("i am the parent process, my processid is %d\n",getpid());
}
fork()
pid=4285
main ()
{
.....
if (pid < 0)
printf("error infork!");
else if (pid == 0)
printf("i am the child process, my processid is %d\n",getpid());
else
printf("i am the parent process, my processid is %d\n",getpid());
}
pid=4286
main ()
{
.....
if (pid < 0)
printf("error infork!");
else if (pid == 0)
printf("i am the child process, my processid is %d\n",getpid());
else
printf("i am the parent process, my processid is %d\n",getpid());
}

小测试

以当前程序所在的进程为根节点的树总共有几个节点（进程）？（包括根节点）

```
#include <stdio.h>
int main(int argc, char* argv[])
{
   …
   fork();
   fork() && fork() || fork();
   …
}
```

Hint：

- *A&&B，如果A==0，不执行&&B；A!=0，继续执行&&B。*
- *A||B，如果A!=0，不执行||B，A==0，继续执行||B。*

小测试

以下面程序所在的进程为根节点的树总共有几个节点（进程）？包括根节点

```
#include <stdio.h>
int main(int argc, char* argv[])
{
   …
   if (fork() && fork() || fork())
   {
      fork();
   }
   …
}
```

Hint：

- *A&&B，如果A==0，不执行&&B；A!=0，继续执行&&B。*
- *A||B，如果A!=0，不执行||B，A==0，继续执行||B。*

Thanks !