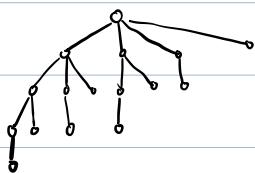


hw2

1. 如图：共有.

$$1+4+6+4+1=16\text{个}$$



2. 由于 execp() 会覆盖子进程原有的代码段内存。

所以 LINE J 并不会被显示

而当 execp() 调用失败时，之后的代码 printf("LINEJ") 可以被执行

3. A: pid = 0

B: pid = 2603

C: pid = 2603

D: pid = 2600.

4 CHILD: 0 -1 -4 -9 -25 // X.

PARENT: 0, 1, 2, 3, 4. // Y

因为：一开始父子进程物理空间相同，但由于“写时复制”技术，子进程改变变量值时，

子进程拥有只属于自己的空间，并完全复制父进程的值，虽然“虚拟地址”值相同，

但父子确实指向不同的物理空间，所以最终 X, Y 行输出不同

5. exec() 函数会把原进程中地址空间代码段替换，由 exec 命令决定调用哪个程序。

如果成功执行 exec(), 则不返回，未执行 LINE:X. 代码。

如果 exec() 执行失败，则函数返回，执行 printf 操作

6. 当子程执行完毕，并调用 exit() 函数后，子进程会释放内存空间等资源。

并保留 PCB 的一些结构，以备父进程调用查看，当此时，在 process table 上，

子进程就处于“terminate process”。

“Terminate State”可以来表达进程已结束 同时发送信号唤醒父进程，

7. "Zombie process" 是当父进程未调用 wait() 函数，但子进程 exit()，已释放大多内存，只保留 PCB 部分内容，记载该进程退出状态等信息的一种状态。

当父进程中调用 wait() 函数，在接收到子进程结束信号后，会结束子进程的 Z 状态，并回收其 PCB 结构。

如果父进程没有调用 wait()，就已经先一步终止，此时，子进程成为“孤儿进程”，由 init 作为其父进程，init 会定期调用 wait()，以便收集 孤儿进程 退出状态，消除 Z 状态。

8. 用例空间：有 Stack：存放函数变量，返回地址，局部变量。

heap：动态内存分配的变量。

data：全局变量。

text：放代码段，程序指令。

内核：进程信息在 PCB 中 (Process Control Block)

包括：进程状态 (new, waiting, running, ready, terminated)

程序计数器。

CPU 寄存器。

CPU 调度信息。

内存管理信息。

记帐信息

I/O 状态信息

9. exec() 与普通函数调用本质的区别是

1. exec() 会把原进程用户空间清空，更换新的程序代码，初始化 data section，重置程序计数器。而函数调用只是调整记帐器内容。

2. exec() 执行完毕后，不会返回原代码（除非调用失败）

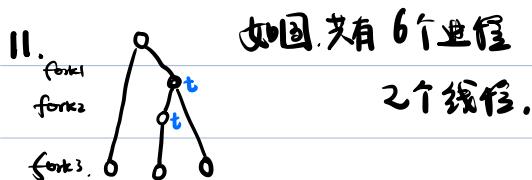
而普通函数执行结束后，会回到返回地址处继续向下执行。

3. exec() 子 用户态与内核态切换，改变，而函数调用不需要。

## 10. 多线程的好处：

多线程并行进行，用户使用系统响应快，资源可以共享，对 CPU 资源的使用更高效，更经济

多线程：共享 b. Heap memory, 动态分配内存 d. Global variables 全局变量。



12. C: CHILD: value = 5 子进程的子线程与父线程共享 Global Var.

P: PARENT: value = 0 父进程与子进程各自有独立空间

13. 普通管道：进程间有父子关系、单向传输、通信完成即关闭

命名管道：进程之间没有必须的关系，半双工传输，有多个进程可以写入，异步式的删除。