- ***实验内容:***在上次 fork 实验的基础上，修改程序，两个子进程分别调用 exec()函数，一个子进程执行系统命令 ps -l -a；另一个子进程执行自己编写的任一程序（例如最简单的在屏幕上打印 hello world 的程序）。父进程则打印一句话：“这是父进程!”

- 实验代码：

```
21      if (pid < 0)
22      {   // fork failed !
23          printf("fork failed!\n");
24          exit(-1);
25      }
26      else if(pid > 0)
27      {   // parent process
28          printf("这是父进程!");          1. 父进程中打印输出
29      }
30      else
31      {   // pid == 0, child process
32          if (i == 0)                    2. 子进程1中执行ps程序
33          {   // 1st child process, 执行ls程序
34              char *argv_list[] = {"ps", "-l", "-a", NULL};
35              int res = execv("/bin/ps",argv_list);
36              if (res < 0)
37                  printf("ls exec error!\n");
38          }
39          else if (i == 1)               3. 子进程2中执行helloworld
40          {   // 2ed child process,  执行helloworld程序
41              char *argv_list[] = {"helloworld", NULL};
42              int res = execv("./helloworld",argv_list);
43              if (res < 0)
44                  printf("helloworld exec error!\n");
45          }
46          else
47          {
48              printf("error!\n");
49              exit(-1);
```

- 实验结果：

```
o2igin@Ubuntu2304:~/Projects/OS-2023Fal/lab2$ ./a.out
hello word!   <-- 子进程2 helloworld程序输出
                                                  子进程1 ps 程序输出
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000    2698    2694  0  80   0 - 76666 do_pol tty2     00:00:00 gnome-session-b
0 S  1000   43522   39676  0  80   0 -   654 do_wai pts/0    00:00:00 a.out
4 R  1000   43523   43522 99  80   0 -  5684 -      pts/0    00:00:00 ps
这是父进程!   <-- 父进程打印输出内容
```

- 实验总结：

本实验通过 fork 创建子进程，而后通过 exec 函数将当前子进程替换为另一个程序继续执行。实验分别通过调用了系统程序 ps 和自定义程序 helloworld，在 shell 得到了输出。

对于 exec 函数，查阅资料后得知 exec 是一个函数族，exec 是这些函数的统称（而不是名为 exec 的函数），在 linux 中包含：execl，execlp，execle，execv，execve 和 execvp，这些函数在执行程序参数设置、环境变量等方面有所不同。