实验一 进程控制

## 实验目的：

* 加深对进程概念的理解，明确进程和程序的区别。
* 掌握Linux系统中的进程创建，管理和删除等操作。
* 熟悉使用Linux下的命令和工具，如man, find, grep, whereis, ps, pgrep, kill, ptree, top, vim, gcc，gdb, 管道|等。

## 基础知识：

* **进程的创建**

Linux中，载入内存并执行程序映像的操作与创建一个新进程的操作是分离的。将程序映像载入内存，并开始运行它，这个过程称为运行一个新的程序，相应的系统调用称为exec系统调用。而创建一个新的进程的系统调用是fork系统调用。

* **exec系统调用**

*#include <unistd.h>*

*int execl (const char \*path, const char \*arg,...);*

execl()将path所指路径的映像载入内存，arg是它的第一个参数。参数可变长。参数列表必须以NULL结尾。

通常execl()不会返回。成功的调用会以跳到新的程序入口点作为结束。发生错误时，execl()返回-1，并设置errno值。

例 编辑/home/kidd/hooks.txt：

*int ret;*

*ret = execl (”/bin/vi”, ”vi”,”/home/kidd/hooks.txt”, NULL);*

*if (ret == -1)*

*perror (”execl”);*

* **fork系统调用**

#include <sys/types.h>

#include <unistd.h>

pid\_t fork (void);

成功调用fork()会创建一个新的进程，它与调用fork()的进程大致相同。发生错误时，fork()返回-1，并设置errno值。

例：

pid\_t pid;

pid = fork ();

if (pid > 0)

printf (”I am the parent of pid=%d!\n”, pid);

else if (!pid)

printf (”I am the baby!\n”);

else if (pid == -1)

perror (”fork”);

* **终止进程**

exit()系统调用：

#include <stdlib.h>

void exit (int status);

* **进程挂起**

pause() 系统调用：

*int pause( void );*

函数pause会把进程挂起，直到接收到信号。在信号接收后，进程会从pause函数中退出，继续运行。

* **wait(等待子进程中断或结束)**

*#include<sys/types.h>*

*#include<sys/wait.h>*

*pid\_t wait (int \* status);*

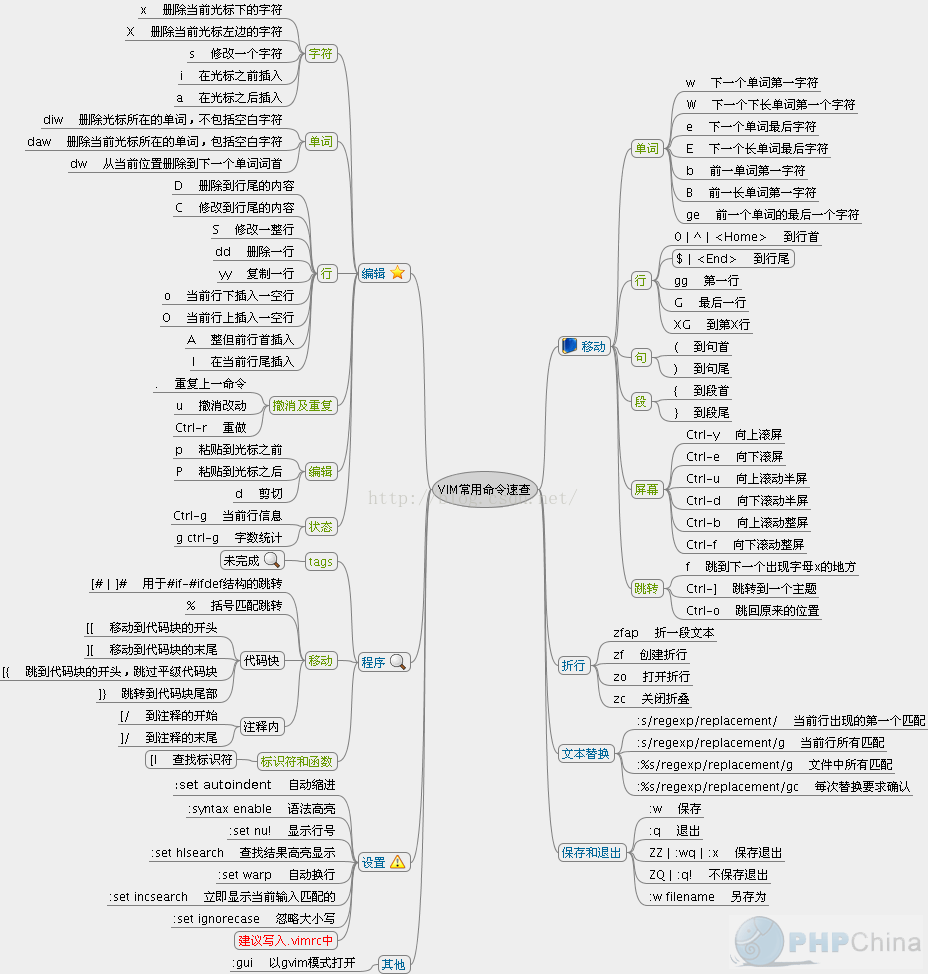
wait()会暂时停止目前进程的执行,直到有信号来到或子进程结束。

如果在调用 wait()时子进程已经结束,则wait()会立即返回子进程结束状态值。

子进程的结束状态值会由参数status返回,而子进程的进程识别码也会一起返回。

如果不在意结束状态值,则参数status可以设成 NULL。

VIM常用命令速查



## 实验题目：

根据课堂所学内容和基础知识介绍，完成实验题目。

* 1、打开一个vi进程。通过ps命令以及选择合适的参数，只显示名字为vi的进程。寻找vi进程的父进程，直到init进程为止。记录过程中所有进程的ID和父进程ID。将得到的进程树和由pstree命令的得到的进程树进行比较。
* 2、编写程序，首先使用fork系统调用，创建子进程。在父进程中继续执行空循环操作；在子进程中调用exec打开vi编辑器。然后在另外一个终端中，通过ps –Al命令、ps aux或者top等命令，查看vi进程及其父进程的运行状态，理解每个参数所表达的意义。选择合适的命令参数，对所有进程按照cpu占用率排序。
* 3、使用fork系统调用，创建如下进程树，并使每个进程输出自己的ID和父进程的ID。观察进程的执行顺序和运行状态的变化。
* 4、修改上述进程树中的进程，使得所有进程都循环输出自己的ID和父进程的ID。然后终止p2进程(分别采用kill -9 、自己正常退出exit()、段错误退出)，观察p1、p3、p4、p5进程的运行状态和其他相关参数有何改变。