

main 函数有 3 个参数：1) argc, 它给出 argv[] 数组中非空指针的数量, 2) argv, 指向 argv[] 数组中的第一个条目, 3) envp, 指向 envp[] 数组中的第一个条目。

Linux 提供了几个函数来操作环境数组：

```
#include <stdlib.h>
```

```
char *getenv(const char *name);
```

返回：若存在则为指向 name 的指针，若无匹配的，则为 NULL。

getenv 函数在环境数组中搜索字符串 “name=value”。如果找到了，它就返回一个指向 value 的指针，否则它就返回 NULL。

```
#include <stdlib.h>
```

```
int setenv(const char *name, const char *newvalue, int overwrite);
```

返回：若成功则为 0，若错误则为 -1。


```
void unsetenv(const char *name);
```

返回：无。

如果环境数组包含一个形如 “name=oldvalue” 的字符串，那么 unsetenv 会删除它，而 setenv 会用 newvalue 代替 oldvalue，但是只有在 overwrite 非零时才会这样。如果 name 不存在，那么 setenv 就把 “name=newvalue” 添加到数组中。

旁注 程序与进程

这是一个适当的地方，停下来，确认一下你理解了程序和进程之间的区别。程序是一堆代码和数据；程序可以作为目标文件存在于磁盘上，或者作为段存在于地址空间中。进程是执行中程序的一个具体的实例；程序总是运行在某个进程的上下文中。如果你想要理解 fork 和 execve 函数，理解这个差异是很重要的。fork 函数在新的子进程中运行相同的程序，新的子进程是父进程的一个复制品。execve 函数在当前进程的上下文中加载并运行一个新的程序。它会覆盖当前进程的地址空间，但并没有创建一个新进程。新的程序仍然有相同的 PID，并且继承了调用 execve 函数时已打开的所有文件描述符。

 **练习题 8.6** 编写一个叫做 myecho 的程序，打印出它的命令行参数和环境变量。

例如：

```
linux> ./myecho arg1 arg2
```

```
Command-line arguments:
```

```
argv[ 0]: myecho
```

```
argv[ 1]: arg1
```

```
argv[ 2]: arg2
```

```
Environment variables:
```

```
envp[ 0]: PWD=/usr0/droh/ics/code/ecf
```

```
envp[ 1]: TERM=emacs
```

```
...
```

```
...
```

```
envp[25]: USER=droh
```

```
envp[26]: SHELL=/usr/local/bin/tcsh
```

```
envp[27]: HOME=/usr0/droh
```