命令行环境

目录

- 命令行参数
- 退出状态
- 环境变量

命令行参数

C 语言程序可以从命令行接收参数。

```
$ ./foo hello world
```

上面示例中，程序 `foo` 接收了两个命令行参数 `hello` 和 `world` 。

程序内部怎么拿到命令行参数呢？ C 语言会把命令行输入的内容，放在一个数组里面。 `main()` 函数的参数可以接收到这个数组。

```
#include <stdio.h>

int main(int argc, char* argv[]) {
  for (int i = 0; i < argc; i++) {
    printf("arg %d: %s\n", i, argv[i]);
  }
}
```

上面示例中， `main()` 函数有两个参数 `argc` （ argument count ）和 `argv` （ argument variable ）。这两个参数的名字可以任意取，但是一般来说，约定俗成就是使用这两个词。

第一个参数 `argc` 是命令行参数的数量，由于程序名也被计算在内，所以严格地说 `argc` 是参数数量 + 1。

第二个参数 `argv` 是一个数组，保存了所有的命令行输入，它的每个成员是一个字符串指针。

以 `./foo hello world` 为例， `argc` 是 3，表示命令行输入有三个组成部分： `./foo` 、 `hello` 、 `world` 。数组 `argv` 用来获取这些输入， `argv[0]` 是程序名 `./foo` ， `argv[1]` 是 `hello` ， `argv[2]` 是 `world` 。一般来说，

argv[1] 到 argv[argc - 1] 依次是命令行的所有参数。 argv[argc] 则是一个空指针 NULL。

由于字符串指针可以看成是字符数组，所以下面两种写法是等价的。

// 写法一
int main(int argc, char* argv[])

// 写法二
int main(int argc, char** argv)

另一方面，每个命令行参数既可以写成数组形式 argv[i]，也可以写成指针形式 *(argv + i)。

利用 argc，可以限定函数只能有多少个参数。

#include <stdio.h>

int main(int argc, char** argv) {
if (argc != 3) {
printf("usage: mult x y\n");
return 1;
}

printf("%d\n", atoi(argv[1]) * atoi(argv[2]));
return 0;
}

上面示例中， argc不等于 3就会报错，这样就限定了程序必须有两个参数，才能运行。

另外， argv数组的最后一个成员是 NULL 指针（ argv[argc] == NULL
）。所以，参数的遍历也可以写成下面这样。

for (char** p = argv; *p != NULL; p++) {
printf("arg: %s\n", *p);
}

上面示例中，指针 p依次移动，指向 argv的每个成员，一旦移到空指针 NULL，就表示遍历结束。由于 argv 的地址是固定的，不能执行自增运算（ argv++），所以必须通过一个中间变量 p，完成遍历操作。

退出状态

C 语言规定，如果 `main()` 函数没有 `return` 语句，那么结束运行的时候，默认会添加一句 `return 0`，即返回整数 `0`。这就是为什么 `main()` 语句通常约定返回一个整数值，并且返回整数 `0` 表示程序运行成功。如果返回非零值，就表示程序运行出了问题。

Bash 的环境变量 `$?` 可以用来读取上一个命令的返回值，从而知道是否运行成功。

```
$ ./foo hello world
$ echo $?
0
```

上面示例中，`echo $?` 用来打印环境变量 `$?` 的值，该值为 `0`，就表示上一条命令运行成功，否则就是运行失败。

注意，只有 `main()` 会默认添加 `return 0`，其他函数都没有这个机制。

环境变量

C 语言提供了 `getenv()` 函数（原型在 `stdlib.h`）用来读取命令行环境变量。

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

int main(void) {
  char* val = getenv("HOME");

  if (val == NULL) {
    printf("Cannot find the HOME environment variable\n");
    return 1;
  }

  printf("Value: %s\n", val);
  return 0;
}
```

上面示例中，`getenv("HOME")` 用来获取命令行的环境变量 `$HOME`，如果这个变量为空（`NULL`），则程序报错返回。

版本号 #2
由 壳壳虫 创建于 2021-09-04 22:34:34 UTC
由 壳壳虫 更新于 2023-04-11 16:48:42 UTC