

乱 着 乐 着

2004-8-30

声明

本文内容仅用于学习及交流，严禁用于以赢利为目的的各种商业行为，否则后果自负。
本文中大部分源码都来自网上，如果相应作者觉得侵犯了您的权益，请通知我，我会立即改正。Email: onlyforos@hotmail.com

乱着乐着

```
main() { printf(&unix["\021%six\012\0"],(unix)["have"]+"fun"-0x60);}
```

1987 年，编写了与 Bourne shell 齐名的 Korn shell 的贝尔实验室 David Korn 提交了如上代码而在当年的国际 C 语言混乱码大赛(IOCCC)中获胜。

不知谁现在就知道这行代码的答案，第一次见是在一个人的签名档中。那阶段正是在四处寻觅深入研究后，膨胀地认为自己已经理解或会用绝大多数 C 的特性。无意中见到这行乍看无从下手代码，至今仍记得当时心中苦涩的失落。：)

本文分析列举几个类似的“混乱”代码，仅供在休闲时刻，博您皱眉一笑。☺

在看懂上面一行代码是如何运行之前，不妨先看一些其他的代码，能否不用运行就知道结果？

```
1. main(){char*p="main(){char*p=%c%s%c;(void)printf(p,34,p,34,10);}%;(void)printf(p,34,p,34,10);}
2. main(a){a="main(a){a=%c%s%c;printf(a,34,a,34);}";printf(a,34,a,34);}
3. main(){char*a="main(){char*a=%c%s%c;int b=%c';printf(a,b,a,b,b);}";int b="";printf(a,b,a,b,b);}
4. main(a){printf(a,34,a="main(a){printf(a,34,a=%c%s%c,34);}";,34);}
```

以上 4 个程序的运行结果是什么？她们的输出就是程序本身。如：

main(a){printf(a,34,a="main(a){printf(a,34,a=%c%s%c,34);}";,34);} 的运行结果就是：

main(a){printf(a,34,a="main(a){printf(a,34,a=%c%s%c,34);}";,34);} 34

在 UltraEdit 中一查 ASCII 码表，34 是”，10 即/012 为/n。稍加考虑，小 CASE 一碟。

再看：

```
1. main(c){putchar(c+'@');c^'?'^%'?main(++c):putchar("\012");}
2. main(l,i){for(i=!l;i<=l;printf("%d\n",l),l+=i,i=l-i);}
```

呵呵，好象看不太出来了。运行一下：

1. 输出 ABCDEFGHIJKLMNOPQRSTUVWXYZ

2. 输出：

1

1

2

3

5

8

13

21

34

55

.....

433494437

701408733

1134903170

1836311903

有名的 Fabonacci 数列。

这两行代码是怎么输出的？怎么退出循环的？

1. 递归调用。‘@’：64。 ‘A’：65。c：编译器初始化为 0.26 个字母。26 为 0x1A，1A^'?'^'?':

```
00011010 ^00111111^00100101 = 0    //'?:0x3F  '%':0x25
```

2. signed int 的取值范围：-2147483648 ~ 2147483647，当 l 值大于 2147483647 时，便上溢为 0，退出循环，程序结束。呵呵，原来上溢也是可以利用的。：)

当然，实际中这种利用最好越少越好了。不仅如此，还要注意不要在无意中发生上溢或下溢的错误。如：

```
unsigned char c;
```

```
for(c = 0; c <= 255; c++){...}
```

就在最后一次循环时 c 累加至 256，可她却没按本意退出循环，而是出现了无限循环。因为此时的 c 已经上溢为 0。

又如：

```
unsigned char i;
```

```
while(--i >= 0){...}
```

当然这又是一个无限循环，因为当 i=0 时而进行最后一次循环时其下溢为无符号数的最大值。

好。下面我们解决本文主题：

```
main() { printf(&unix["\021%six\012\0"],(unix)["have"]+"fun"-0x60);}
```

我们刚开始学习 C 语言时便知道了怎么使用 printf，上面的这个 printf 怎么好象不符合我们习惯的：

printf(“%s\n”,“abcde”);?

1. 首先，看看 printf 中的前半句：&unix[“\021%six\012\0”]。

1) unix

没看到哪儿声明定义 unix，怎么就可以直接拿来使用呢？在 linux 下做如下操作：

```
<yug>[/home/yug/tmp/learn]%cpp -v
```

```
Reading specs from /usr/lib/gcc-lib/i386-redhat-linux/3.2.2/specs
```

```
Configured with: ../configure --prefix=/usr --mandir=/usr/share/man --infodir=/u
```

```
sr/share/info --enable-shared --enable-threads=posix --disable-checking --with-s
```

```
ystem-zlib --enable-__cxa_atexit --host=i386-redhat-linux
```

```
Thread model: posix
```

```
gcc version 3.2.2 20030222 (Red Hat Linux 3.2.2-5)
```

```
/usr/lib/gcc-lib/i386-redhat-linux/3.2.2/cpp0 -lang-c -v -D__ELF__ -Dunix -D__g
```

```
nu_linux__ -Dlinux -D__ELF__ -D__unix__ -D__gnu_linux__ -D__linux__ -D__unix -D__
```

```
_linux -Asystem=posix -D__NO_INLINE__ -D__STDC_HOSTED__=1 -Acpu=i386
```

```
-Amachine=i
```

```
386 -Di386 -D__i386 -D__i386__ -D__tune_i386__ -
```

```
GNU CPP version 3.2.2 20030222 (Red Hat Linux 3.2.2-5) (cpplib) (i386 Linux/ELF)
```

-D 在预编译器 CPP 中定义宏，且将值设为 1。从上面信息我们可以看到'-Dunix'。宏 unix 的值为 1。

如果想知道系统的预定义宏都有哪些，可进行如下操作：

```
<yug>[/home/yug]%touch foo.h //假设没有 foo.h 这个文件
```

```
<yug>[/home/yug]%cpp -dM foo.h
```

```
#define __HAVE_BUILTIN_SETJMP__ 1
#define __unix__ 1
#define unix 1
#define __i386__ 1
#define __SIZE_TYPE__ unsigned int
#define __ELF__ 1
#define __linux 1
#define __unix 1
#define __linux__ 1
#define __USER_LABEL_PREFIX__
#define linux 1
#define __STDC_HOSTED__ 1
#define __WCHAR_TYPE__ long int
#define __gnu_linux__ 1
#define __WINT_TYPE__ unsigned int
#define i386 1
#define __STDC__ 1
#define __PTRDIFF_TYPE__ int
#define __tune_i386__ 1
#define __REGISTER_PREFIX__
#define __NO_INLINE__ 1
#define __i386 1
#define __VERSION__ "3.2.2 20030222 (Red Hat Linux 3.2.2-5)"
```

不同的 CPU，不同的 OS，有不同的预定义宏。以上宏的出处也许可从如下文件中查找(不一定对，仅供参考!)

```
gcc-3.3.2\gcc\config\i386\linux.h(81): builtin_define_std ("unix"); \
```

通过上面努力，可知道宏‘unix’的值为 1。

2) &(1)[“\021%six\012\0”]

- a). char str[1];
- b). char str[]="abcde";
str[1]; //’b’
- c). “abcde”[1]; //’b’

以上三种表示都是大家熟悉的数组。数组名及字符串表示一个首地址，str[1]即是数组 str 的首地址与下标 1 相加后的地址单元中的值(这么简单的东东居然说的这么复杂：0)。

那么上面(b)(c)中，也应该很容易理解下面的几种表示是等价的。

```
str[1] ⇔ “abcde”[1];
str+1 ⇔ &”abcde”[1] ⇔ “bcde”;
```

另外，既然是相加，对于编译器而言，’a+b’与’b+a’完全相同。所以数组’str[1]’与’1[str]’两种表示的结

果完全相同。同样的道理，如下的表示也是完全等价的。

```
"abcde"[1] ⇔ 1["abcde"];
"abcde"+1 ⇔ &1["abcde"] ⇔ "bcde";
```

最后，如前所说'012'即'n'，'\0'没什么具体的用可能是用于迷惑或凑数。

所以，对于'&unix["\021%six\012\0"]'：

```
&unix["\021%six\012\0"] =>
&1["\021%six\012\0"]   =>
&"\021%six\012\0"[1]   =>
"%six\n"
```

2. 接着，分析后半句：'(unix)["have"]+"fun"-0x60'

有了上面的基础，这句就有点一目了然了。

```
(unix)["have"]+"fun"-0x60 =>
(1)["have"]+"fun"-0x60   =>
"have"[1]+"fun"-0x60     =>
'a'+"fun"-0x60           =>
0x61+"fun"-0x60          =>
"fun"+1                   =>
"un"
```

那么，小强，main(){printf("%six\n","un");}的结果是~~：

```
unix
```

呵呵，是不是意犹未尽啊(不知有几人能耐心地看到这里：))。下面再列举几行代码，并附上运行结果。不过这些我没研究或没研究明白，有谁知道是怎么运行的，给我来封信 yu.gen@zte.com.cn，不胜感激。

1. 45 个字符计算某天是星期几的一行代码。

Calculate the day of the week in 45 characters of code. Written by [Mike Keith](#).

```
(d+=m<3?y--:y-2,23*m/9+d+4+y/4-y/100+y/400)%7
```

```
<yug>[/home/yug/tmp/learn/obfuscated]%cat week.c
```

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

```
int main()
```

```
{
```

```
    int y,m,d;
```

```
    static char* strdaynames = "SunMonTueWedThuFriSat";
```

```
    printf("the year is:\n");
```

```
    scanf("%d",&y);
```

```
    printf("the month is:\n");
```

```
    scanf("%d",&m);
```

```
    printf("the day is:\n");
```

```
    scanf("%d",&d);
```

```

d=(d+=m<3?y--:y-2,23*m/9+d+4+y/4-y/100+y/400)%7;
printf("the day is [%s]!\n",strdaynames+d*3);
return 0;
}

```

```

<yug>[/home/yug/tmp/learn/obfuscated]%!g
gcc -o week week.c
<yug>[/home/yug/tmp/learn/obfuscated]%%week
the year is:
2004
the month is:
8
the day is:
28
the day is [Sat]!

```

2. 160 个字符计算 π 小数点的 800 位:

Calculate pi to 800 digits in 160 characters of code. Written by Dik T. Winter at CWI.

```

int a=10000,b,c=2800,d,e,f[2801],g;main(){for(;b-c;)f[b++]=a/5;
for(;d=0,g=c*2;c-=14,printf("%.4d",e+d/a),e=d%a)for(b=c;d+=f[b]*a,
f[b]=d%--g,d/=g--,--b;d*=b);}

```

```

<yug>[/home/yug/tmp/learn/obfuscated]%%gcc -o 314 314.c
<yug>[/home/yug/tmp/learn/obfuscated]%%314
31415926535897932384626433832795028841971693993751058209749445923078164062862089986280348253
42117067982148086513282306647093844609550582231725359408128481117450284102701938521105559644
62294895493038196442881097566593344612847564823378678316527120190914564856692346034861045432
66482133936072602491412737245870066063155881748815209209628292540917153643678925903600113305
30548820466521384146951941511609433057270365759591953092186117381932611793105118548074462379
96274956735188575272489122793818301194912983367336244065664308602139494639522473719070217986
09437027705392171762931767523846748184676694051320005681271452635608277857713427577896091736
37178721468440901224953430146549585371050792279689258923542019956112129021960864034418159813
6297747713099605187072113499999983729780499510597317328160963185

```

3.

```

int main() {
    unsigned char x[] = "31C031DB31D2B00443E800000005983C10FB233CD8"
"031C031DB40CD804469656E737461672C2030362E30312E204368616F735472"
"65666620696D204B6165757A6368656E2061622031393A30302E0A8B",i=0,j;
    while(x[i]){j=x[i+2];x[i+2]=0;x[i/2]=strtol(x+i,0,16);x[i+=2]=j;}
    return ((int(*)())x)();
}

```

```

<yug>[/home/yug/tmp/learn/obfuscated]%%cli1
Dienstag, 06.01. ChaosTreff im Kaeuzchen ab 19:00.

```

4.

```
int main(){char*a="7EE977777777779777779C9C7CC7CC7ECEEC1=EG;E79CC9E7EG"
";CCE9E89P9O==O==O=CKE86OO7O9:89OP9:=EOEQD9.;G;C;;;;;OOOSCI5;EG;;;G;"
";G.E.;G;;7E=9O9OOOQOOS;G;EON:EEQOQREQREDQE@EQO:EGG;G7;GGGGGFEDDEE=0",*b=
"\n/_\\";while(*a){putchar(b[(/*a-'0')/6]);putchar(b[(/*a++-48)%6]);}}
```

```
<yug>[/home/yug/tmp/learn/obfuscated]%cli2
```

/ _ | | _ _ _ _ | | _ _ _ / _ / _ / / / / / _ \ _ /
 | | | | _ \ / _ | / _ \ _ | _ | _ / _ \ | | | | | | | | _ \
 | | _ | | | | | | | _ \ _ \ _ | | | _ / _ | | | | | | | | _ |
 \ _ | | | | \ _ | \ _ / _ \ _ | | | | | | | | | | | | \ _ / _ /

5.

```
#include <stdio.h>

main(t,_a) char *a; { return!0<t?t<3?main(-79,-13,a+main(-87,1,_a,main(-86,0,a+1)+a)):1,t<_?main(t+1,_a):3,main(-94,-27+t,a)&&t==2?_<13?main(2,_+1,"%s %d %d\n"):9:16:t<0?t<-72?main(_t,

"@n'+,#'/* {} w+/w#cdnr/+, {} r/*de}+,*{*+/,w{%/w#q#n+,#{1+/,n{n+/,+ #n+/,#n+/,#q#n+/,+k#;*,+/'r :d*3,} {w+K w'K:'+}e#;dq#l \

q#'+d'K#!/+k#;q#r}eKK#}w'r}eKK{nl}'/#.#q#n'}){#}w'}){nl}'/+ #n';d}rw' i;# \

){nl}'/n{n#'; r{#w'r nc{nl}'/#{1,+ 'K {rw' iK{;[{nl}'/w#q#n'wk nw' \

iwk{KK{nl}'/w{%l##w#'} i; :{nl}'/*{q#ld;r'}{nlwb!/*de}'c \

;:{nl}-{rw}'/+,}##*'}#nc,'#nw}'/kd'+e}+;#rdq#w! nr/' ) }+} {rl#}'n')# \

}'+}##(!/!):t<-50?_==*a?putchar(31[a]):main(-65,_a+1):

main(( '*a==/' )+t,_a+1):0<t?main(2,2,"%s"):*a==/'||main(0,main(-61,*a,

"!ek;dc i@bK('q)-[w]*%n+r3#l, } \:nuwloca-O;m .vpbks,fxntdCeghry"),a+1); }
```

```
<yug>[/home/yug/tmp/learn/obfuscated]%a.out
```

On the first day of Christmas my true love gave to me
a partridge in a pear tree.

On the second day of Christmas my true love gave to me
two turtle doves
and a partridge in a pear tree.

On the third day of Christmas my true love gave to me
three french hens, two turtle doves
and a partridge in a pear tree.

On the fourth day of Christmas my true love gave to me
four calling birds, three french hens, two turtle doves
and a partridge in a pear tree.

On the fifth day of Christmas my true love gave to me
five gold rings;
four calling birds, three french hens, two turtle doves
and a partridge in a pear tree.

On the sixth day of Christmas my true love gave to me
six geese a-laying, five gold rings;
four calling birds, three french hens, two turtle doves
and a partridge in a pear tree.

On the seventh day of Christmas my true love gave to me
seven swans a-swimming,
six geese a-laying, five gold rings;
four calling birds, three french hens, two turtle doves
and a partridge in a pear tree.

On the eighth day of Christmas my true love gave to me
eight maids a-milking, seven swans a-swimming,
six geese a-laying, five gold rings;
four calling birds, three french hens, two turtle doves
and a partridge in a pear tree.

On the ninth day of Christmas my true love gave to me
nine ladies dancing, eight maids a-milking, seven swans a-swimming,
six geese a-laying, five gold rings;
four calling birds, three french hens, two turtle doves
and a partridge in a pear tree.

On the tenth day of Christmas my true love gave to me
ten lords a-leaping,
nine ladies dancing, eight maids a-milking, seven swans a-swimming,
six geese a-laying, five gold rings;
four calling birds, three french hens, two turtle doves
and a partridge in a pear tree.

On the eleventh day of Christmas my true love gave to me
eleven pipers piping, ten lords a-leaping,
nine ladies dancing, eight maids a-milking, seven swans a-swimming,
six geese a-laying, five gold rings;
four calling birds, three french hens, two turtle doves
and a partridge in a pear tree.

On the twelfth day of Christmas my true love gave to me

twelve drummers drumming, eleven pipers piping, ten lords a-leaping,
nine ladies dancing, eight maids a-milking, seven swans a-swimming,
six geese a-laying, five gold rings;
four calling birds, three french hens, two turtle doves
and a partridge in a pear tree.

6.

```
main(){char *b=".:;|>")&IH%*#" ;float i,j,k,r,x,y=-16;while
(puts(""))y++<15)for(x=0;x++<84;putchar(b[(int)k&15]))for(i=k
=r=0;j=r*r-i-2+x/25,i=2*r*i+y/10,j*j+i*k<11&&k++<111;r=j);}
```

```
<yug>[/home/yug/tmp/learn/obfuscated]%aca
```

```
<yug>[ /home/yug/tmp/learn/obfuscated]%
```

注：本文列举代码均在 RedHat Linux9.0 上运行通过，gcc 版本如下：

```
<yug>[/home/yug/tmp/learn/obfuscated]%gcc --version
```

gcc (GCC) 3.2.2 20030222 (Red Hat Linux 3.2.2-5)

Copyright (C) 2002 Free Software Foundation, Inc.

This is free software; see the source for copying conditions. There is NO

warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

附：详解

```
main() { printf(&unix["\021%six\012\0"],(unix)["have"]+"fun"-0x60);};
```

```
/* ioccc.c */
```

```
/* IOCCC best one-liner winner 1987 by David Korn ---
```

```
main() { printf(&unix["\021%six\012\0"],(unix)["have"]+"fun"-0x60);};
```

```
from <http://www.ioccc.org/years.html#1987>
```

```
*/
```

```
/* A detailed set of samples to show how this works
```

```
by David Ireland, copyright (C) 2002.
```

```
*/
```

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

```
main()
```

```
{
```

```
    /* OK, 'unix' is an int by default.
```

```
    You may need to declare with fussy compilers (MSVC),
```

```
    and perhaps explicitly not with others (gcc).
```

```
    */
```

```
    //int unix;
```

```
    printf("unix=%d\n", unix); /* =1 */
```

```
    /* Now why does this variable have the value one? */
```

```
    /* A. Because the first int declared in main is 'argc' in K&R C,
```

```
    as in main(int argc, char *argv[]).
```

```
    It will be set to one when main() is called by the OS
```

```
    - try adding extra args to cmd line and see it increase.
```

```
(Or try declaring another variable before it like this:
    int dummy;
    int unix;
    What value will 'unix' have now?)
*/

/* This prints the string "un",
   i.e. "fun" starting at offset [1] */
printf("%s\n", "fun"+1);

/* This prints 97 = the int value of the 2nd char 'a' */
printf("%d\n", "have"[1]);

/* just like this */
printf("%d\n", 'a');

/* ditto because x[1] = 1[x] */
printf("%d\n", (1)["have"]);

/* 97 - 96 = 0x61 - 0x60 = 1 */
printf("%d\n", (1)["have"] - 0x60);

/* So this is the same as "fun" + 1, printing "un" */
printf("%s\n", "fun" + ((1)["have"] - 0x60));

/* Rearrange and use unix variable instead of 1 */
printf("%s\n", (unix)["have"]+"fun"-0x60);

/* ...thus we have the first argument in the printf call. */

/* Both these print the string "bcde", ignoring the 'a' */
printf("%s\n", "abcde" + 1);
printf("%s\n", &"abcde"[1]);

/* so does this */
printf("%s\n", &(1)["abcde"]);

/* and so does this (NB [] binds closer than &) */
printf("%s\n", &unix["abcde"]);

/* This prints "%six" + newline */
printf("%s", &"?%six\n"[1]);

/* So does this: note that
```

```
\012 = 0x0a = \n,
the first \021 char is ignored,
and the \0 is superfluous, probably just for symmetry */
printf("%s", &"\021%six\012\0"[1]);

/* and so does this */
printf("%s", &unix["\021%six\012\0"]);

/* Using this as a format string, we can print "ABix" */
printf(&unix["\021%six\012\0"], "AB");

/* just like this does */
printf("%six\n", "AB");

/* So, we can print "unix" like this */
printf("%six\n", (unix)["have"]+"fun"-0x60);

/* or, finally, like this */
printf(&unix["\021%six\012\0"],(unix)["have"]+"fun"-0x60);

return 0;
}

/* Korn winner, rearranged for fussier compliers ---
#include <stdio.h>
int main()
{
    int unix;
    printf(&unix["\021%six\012\0"],(unix)["have"]+"fun"-0x60);
    return 0;
}
*/
```