Common C Programming Errors

```
int x = 2;
switch(x) {
case 2:
  printf("Two\n");
case 3:
  printf("Three\n");
}
```

Giorgio Richelli
giorgio_richelli@it.ibm.com

Common C Programming Errors

```
int x = 5;
if ( x = 6 )
  printf("x equals 6\n");
```

Giorgio Richelli
giorgio_richelli@it.ibm.com

Common C Programming Errors

```
int x;
char *st = malloc(31);
double dbl;

scanf("%d", &x);
scanf("%30s", &st);
scanf(“%f”,&dbl);
```

Giorgio Richelli
giorgio_richelli@it.ibm.com

Common C Programming Errors

```
int i=0;
char a[1000];
get_data(a);
while (a[i++]!='\n') {
 do_something();
 }
```

Giorgio Richelli
giorgio_richelli@it.ibm.com

Common C Programming Errors

```
#include <string.h>
int main()
{
  char *st;
  strcpy(st, "abc");
  return 0;
}
```

Giorgio Richelli
giorgio_richelli@it.ibm.com

Common C Programming Errors

```
#include <string.h>
char * foo()
{
  char st[20];
  strcpy(st, "abc");
  return(st);
}
```

Giorgio Richelli
giorgio_richelli@it.ibm.com

Common C Programming Errors

```
#include <string.h>
char * foo()
{
  char *st=malloc(20);
  strcpy(st, "abc");
  return(st);
}
```

Giorgio Richelli
giorgio_richelli@it.ibm.com

Common C Programming Errors

```
#include <string.h>
char * foo()
{
  char *st;
  if (!(st=malloc(20)) return(NULL);
  strcpy(st, "abc");
  return(st);
}
```

Giorgio Richelli
giorgio_richelli@it.ibm.com

Common C Programming Errors

```
char ch = 'A';
char ch = "A";

const char * st = "A";
const char * st = 'A';
```

Giorgio Richelli
giorgio_richelli@it.ibm.com

Common C Programming Errors

```
char st1[] = "abc";
char st2[] = "abc";
if ( st1 == st2 )
  printf("Yes");
else
  printf("No");
```

Giorgio Richelli
giorgio_richelli@it.ibm.com

Common C Programming Errors

```
char st1[] = "abc";
char st2[] = "abc";
if ( !strcmp(st1,st2 ))
  printf("Yes");
else
  printf("No");
```

Giorgio Richelli
giorgio_richelli@it.ibm.com

Common C Programming Errors

```
char st1[] = "abcd";
char st2[10] ;
strncpy(st2,st1,3);
do_something(st2);
```

Giorgio Richelli
giorgio_richelli@it.ibm.com

Common C Programming Errors

```
char * copy_str = malloc( strlen(orig_str));
strcpy(copy_str, orig_str);
```

Giorgio Richelli
giorgio_richelli@it.ibm.com

Common C Programming Errors

```
int count_line_size( FILE * fp )
{
  char ch;
  int  cnt = 0;
  while( (ch = fgetc(fp)) != EOF && ch != '\n')
    cnt++;
  return cnt;
}
```

Giorgio Richelli
giorgio_richelli@it.ibm.com

Common C Programming Errors

```
int count_line_size( FILE * fp )
{
  char ch;
  int  cnt = 0;
  while((int)(ch=(char)fgetc(fp))!=EOF&&ch!='\n')
        cnt++;
  return cnt;
}
```

Giorgio Richelli
giorgio_richelli@it.ibm.com

Common C Programming Errors

```
FILE * fp = fopen("test.txt", "r");
 char line[100];
 while( ! feof(fp) ) {
   fgets(line, sizeof(line), fp);
   fputs(line, stdout);
 }
 fclose(fp);
```

Giorgio Richelli
giorgio_richelli@it.ibm.com

Common C Programming Errors

```
FILE * fp = fopen("test.txt", "r");
 char line[100];
 while( 1 ) {
   fgets(line, sizeof(line), fp);
   if ( feof(fp) )  break;
   fputs(line, stdout);
 }
 fclose(fp);
```

Giorgio Richelli
giorgio_richelli@it.ibm.com

Common C Programming Errors

```
FILE * fp = fopen("test.txt", "r");
 char line[100];
 while( fgets(line, sizeof(line), fp) != NULL )
   fputs(line, stdout);
 fclose(fp);
```

Giorgio Richelli
giorgio_richelli@it.ibm.com

Common C Programming Errors

```
 if( 0 < a < 5) c;
 if( a =! b) c;
int foo (a)
    { if (a) return(1); }
foo(pointer->member, pointer = &buffer[0]);
```

Giorgio Richelli
giorgio_richelli@it.ibm.com

Robust Programming

```
int foo( char *c, int p1, int p2 )
{
  if ( c == NULL ) {
    printf("error null object\n");
    return .. ; /* an appropriate error value */
    }
  ... /* perform something on c */
  }
```

Giorgio Richelli
giorgio_richelli@it.ibm.com

Robust Programming

```
#include <assert.h>
int foo(char *c, int p1, int p2 )
{
  assert( c != NULL );

  ... /* perform something on c */
  }
```

Giorgio Richelli
giorgio_richelli@it.ibm.com

Other Common Errors

- Not checking *malloc()* return value
- Calling *free()* multiple times on the same pointer
- Calling *free()* on a memory area not allocated by *malloc/calloc/realloc*
- Using a *freed* pointer
- *Memory Leaks*

Giorgio Richelli
giorgio_richelli@it.ibm.com

Hints & Tips

- Use *indent*
- Initialize variables (!)
- Always check:
  - input parameters (range, NULL, etc..)
  - return values
- Return Sensible Values to Caller
  - never use void functions
  - a global variable / structure with error information

Giorgio Richelli
giorgio_richelli@it.ibm.com

Hints & Tips

- Use *assert* or similar constructs
- Preprocessor & Conditional Compilation
  - #if defined (DEBUG) ... #endif
- Cleanup & Exit, instead of unwinding stacks
  - *atexit()*
- Exploit debugger
  - cc -g ....
  - gdb

Giorgio Richelli
giorgio_richelli@it.ibm.com