Shooting Yourself in the Foot

It has been said of the C programming language, that you can shoot yourself in the foot. The same is also true of other programming languages but different…

370 JCL - You send your foot down to MIS with a 4000-page document explaining how you want it to be shot. Three years later, your foot comes back deep-fried.

6502 - You shoot yourself in the foot.

68000 - You can't decide which gun and which bullet to use, so you hang yourself.

8080 - You foot yourself in the shoot.

80x86 - The gun isn't in the same segment as your feet, so you can't shoot them.

Access - You try to point the gun at your foot, but it shoots holes in all your Borland distribution diskettes instead.

Ada
1 - If you are dumb enough to actually use this language, the United States Department of Defense will kidnap you, stand you up in front of a firing squad, and tell the soldiers, "Shoot at his feet."

2 - After correctly packaging your foot, you attempt to concurrently load the gun, pull the trigger, scream and shoot yourself in the foot. When you try, however, you discover that your foot is of the wrong type.

3 After correctly packing your foot, you attempt to concurrently load the gun, pull the trigger, scream, and confidently aim at your foot knowing it is safe. However the cordite in the round does an Unchecked Conversion, fires and shoots you in the foot anyway.

4 - You scour all 154e56 pages of the manuals, looking for references to "foot", "leg", or "toes"; then get hopelessly confused and give up. You sneak in when the boss isn't around and actually write the stinkin' thing in C, and turn 7689 pages of source code in to the review committee, knowing that they'll never look at it. When the program needs maintenance, you resign.

Algol - You shoot yourself in the foot with a musket. The musket is aesthetically fascinating, and the wound baffles the adolescent medic in the emergency room.

Algol 60 - You spend hours trying to figure out how to fire the gun since it doesn't have any provision for input or output.

Algol 68 - You mildly deprocedure the gun, the bullet gets firmly de-referenced, and your foot is

strongly coerced to void.

Amos - ShootInFoot LEFT
>F1<

APL
1 - You shoot yourself in the foot, then spend all day figuring out how to do it in fewer characters.

2 - You hear a gunshot and there's a hole in your foot, but you don't remember enough linear algebra to understand what has happened.

3 - @#&^$%&%^ foot

ASP - You try to shoot yourself in the foot, however the most advanced thing you can manage is to cut your wrist.

Assembler
1 - You try to shoot yourself in the foot, only to discover you must first invent the gun, the bullet, the trigger, and your foot.

2 - You crash the OS and overwrite the root disk. The system administrator arrives and shoots you in the foot. After a moment of contemplation, the administrator shoots himself in the foot and then hops around the room rabidly shooting at everyone in sight.

3 - By the time you've written the gun, you are dead, and don't have to worry about shooting your feet. Alternatively, you shoot and miss, but don't notice.

4 - Using only 7 bytes of code, you blow off your entire leg in only 2 CPU clock ticks.

BASIC (compiled) - You shoot yourself in the foot with a BB using a SCUD missle launcher.

BASIC (interpreted)
1 - Shoot yourself in foot with water pistol. On big systems, continue until entire lower body is waterlogged.

2 - Lacking a gun, you hold the bullet in your hand and throw it at your foot....and miss.

C - You shoot yourself in the foot.

C++ - You accidentally create a dozen instances of yourself and shoot them all in the foot. Providing emergency medical assistance is impossible since you can't tell which are bitwise copies and which are just pointing at others and saying, "That's me, over there."

C# - Copy how Java shot itself in the foot. Explain how you did it better.

Clipper - You grab a bullet, get ready to insert it in the gun so that you can shoot yourself in the foot, and discover that the gun that the bullet fits has not yet been built, but should be arriving in the mail REAL SOON NOW.

COBOL
1 - USEing a COLT 45 HANDGUN, AIM gun at LEG.FOOT, THEN place ARM.HAND.FINGER on HANDGUN.TRIGGER and SQUEEZE. THEN return HANDGUN to HOLSTER. CHECK whether shoelace needs to be retied.

2 - Allocate $500000 for the project. Define gun,bullet,foot. Run press_trigger. Go for coffee break. Return in time to put foot under bullet.

3 USE HANDGUN.COLT(45), AIM AT LEG.FOOT THEN WITH ARM.HAND.FINGER ON HANDGUN.COLT(TRIGGER) PERFORM SQUEEZE, RETURN HANDGUN.COLT TO HIP.HOLSTER, SCREAM.

4 - You try to shoot yourself in the foot, but the gun won't fire unless it's aligned in column 8.

Concurrent Euclid - You shoot yourself in somebody else's foot.

CP/M - You remember when shooting yourself in the foot with a BB gun was a big deal.

CSS - You shoot your right foot with one hand, then switch hands to shoot your left foot but you realize that the gun has turned into a banana.

dBase
1 - You buy a gun. Bullets are only available from another company and are promised to work so you buy them. Then you find out that the next version of the gun is the one that is scheduled to actually shoot bullets.

2 - You squeeze the trigger, but somebody corrupted the index and the bullet shoots in your eye.

3 - You squeeze the trigger, but the bullet moves so slowly that by the time your foot feels the pain you've forgotten why you shot yourself anyway.

dBase IV v1.0 - You pull the trigger, but it turns out that the gun was a poorly designed grenade, and the whole building blows up.

DCL
```
 $ MOUNT/DENSITY=.45/LABEL=BULLET/MESSAGE="BYE" BULLET::BULLET$GUN
SYS$BULLET
 $ SET GUN/LOAD/SAFETY=OFF/SIGHT=NONE/HAND=LEFT/CHAMBER=1
   /ACTION=AUTOMATIC/LOG/ALL/FULL SYS$GUN_3$DUA3:[000000]GUN.GNU
 $ SHOOT/LOG/AUTO SYS$GUN SYS$SYSTEM:[FOOT]FOOT.FOOT

 %DCL-W-ACTIMAGE, error activating image GUN
```

-CLI-E-IMGNAME, iamge file 3DUA240:[GUN]GUN.EXE;1
-IMGACT-F-NOTNATIVE, image is not an OpenVMS Alpha AXP image

DOS - You finally find the gun, but can't find the file with the foot for the life of you.

Eiffel
1 - You create a GUN object, two FOOT objects and a BULLET object. The GUN passes both the FOOT objects a reference to the BULLET. The FOOT objects increment their hole counts and forget about the BULLET. A little demon then drives a garbage truck over your feet and grabs the bullet (both of it) on the way.

2 - You take out a contract on your foot. The precondition is that there's a bullet in the gun, the postcondition is that there's a hole in your foot.

English - You put your foot in your mouth, then bite it off.

FidoNet - You put your foot in your mouth, then echo it internationally.

forth
1 - Foot in yourself shoot.

2 - First you decide to leave the number of toes lost on the stack and then implement the word foot-toes@ which takes three numbers from the stack: foot number, range, and projectile mass (in slugs) and changes the current vocabulary to 'blue'. While testing this word you are arrested by the police for mooning (remember, this is a bottom-up language) who demonstrate the far better top-down approach to damaging yourself.

3 - BULLET DUP3 * GUN LOAD FOOT AIM TRIGGER PULL BANG! EMIT DEAD IF DROP ROT THEN (This takes about five bytes of memory, executes in two to ten clock cycles on any processor and can be used to replace any existing function of the language as well as in any future words). (Welcome to bottom up programming - where you, too, can perform compiler pre-processing instead of writing code)

FORTRAN
1 - You shoot yourself in each toe, iteratively, until you run out of toes, then you read in the next foot and repeat. If you run out of bullets, you continue anyway because you have no exception-handling ability.

2 You shoot yourself in each toe, iteratively, until you run out of toes; then you shoot the sixth bullet anyway since no exception processing was anticipated.

Genetic Algorithms - You create 10,000 strings describing the best way to shoot yourself in the foot. By the time the program produces the optimal solution, humans have evolved wings and the problem is moot.

Haskell - Shooting yourself in the foot is considered a side effect so you try to use monads to

handle the bullet's I/O but find you're not smart enough.

HTML
1 - You shoot yourself in the foot, only to find out that no matter how gory the result looks, your foot keeps working. Your foot finally stops working when you stub your toe kicking the box the gun came in.

2 - Shoot here

HyperTalk
1 - Put the first bullet of the gun into foot left of leg of you. Answer the result.

2 - You describe how to shoot yourself in the foot, which not only happens, but you also get cool visual effects.

Java
1 - You write a program to shoot yourself in the foot and put it on the Internet. People all over the world shoot themselves in the foot.

2 - The gun fires just fine, but your foot can't figure out what the bullets are and ignores them.

Javascript

1 - You've perfected a robust, rich user experience for shooting yourself in the foot. You then find that bullets are disabled on your gun.

2 - You pull the trigger, wait 10 minutes, look down the barrel, and then the gun shoots you in the face.

Lisp - You shoot yourself in the appendage which holds the gun with which you shoot yourself in the appendage which holds the gun with which you shoot yourself in the appendage which holds the gun with which you shoot yourself in the appendage which holds ...

Logo
1 - You can easily shoot the gun, but first you have to work out the geometry to be sure the bullet goes into your foot.

2 - You tell a turtle to draw a picture of a foot and a gun, then shoot the turtle.

MED-PC - You can't shoot yourself in the foot, but you can train a pigeon to pull the trigger.

Microsoft C++ w/Windows SDK - You write about 100 lines of code to print "hello, world!" in a dialogue box, only to have a UAE pop up when you click on OK. This shuts down the program manager, leaving you nothing but a screensaver. You then fly to Washington and shoot Bill Gates in the foot.

Modula-2 - After realizing that you can't actually accomplish anything in this language, you shoot yourself in the head.

MOO - You ask a wizard for a pair of hands. After lovingly handcrafting the gun and each bullet, you tell everyone that you've shot yourself in the foot.

Motif - You spend days writing a UIL description of your foot, the trajectory, the bullet, and the intricate scrollwork on the ivory handles of the gun. When you finally get around to pulling the trigger, the gun jams.

Nedula/2 - After realizing that you can't actually accomplish anything in this language, you shoot yourself in the head.

.Net - Microsoft hands you a gun and swears blind it's a toenail clipper. Somebody throws a chair at you.

Objective-C - You write a protocol for shooting yourself in the foot so that all people can get shot in their feet.

Occam - You shoot both your feet with several guns at once.

Oracle - You decide to shoot yourself in the foot and go out to buy a gun - except the gun won't work without "deploying" a shoulder holster solution, and relational titanium alloy bullets, and body armor infrastructure, and a laser sight assistant, and a retractable arm stock application, and an enterprise team of ballistics experts and a chiropodist.

ORCA/C - Byteworks keeps promising to supply good ammunition REAL SOON NOW.

Paradox - Not only can you shoot yourself in the foot, your users can too.

Pascal - The compiler won't let you shoot yourself in the foot.

Perl
1 - !($foot =-/left/) # ! read as "Bang!"

2 - You separate the bullet from the gun with a hyperoptimized regexp, and then you transport it to your foot using an array of arrays of arrays. However, the program fails to run and you can't correct it since you don't understand what the heck it is you've written.

3 - You shoot yourself in the foot, but nobody can understand how you did it. Six months later, neither can you.

PHP -
1 - If you're lucky and the HTTP connection doesn't time out, you shoot yourself in the foot.

2 - You shoot yourself in the foot with a gun made from pieces from 300 other guns.

PL/1 - After consuming all system resources including bullets, the data processing department doubles its size, acquires two new mainframes and drops the original on your foot.

Prolog - You tell your program you want to be shot in the foot. The program figures out how to do it, but the syntax doesn't allow it to explain.

Prolog (interpreted) - Your program tries to shoot you in the foot, but you die of old age before the bullet leaves the gun.

Python - You try to shoot yourself in the foot but you just keep hitting the whitespace between your toes.

Revelation - You'll be able to shoot yourself in the foot just as soon as you figure out what all these bullets are for.

Ruby on Rails - You start to think about shooting yourself in the foot only to discover tat someone has already shot your foot for you.

SAS
1 - You spend three hours trying to cut your way through your foot with a rock flake, only to realize that the language was invented before guns allowed you to shoot yourself in the foot interactively in one easy step with no programming.

2 - You have no idea that the gun, the bullet, or your foot exists. The gun is locked in a safe in a bank vault on the other side of the galaxy, the bullet is locked in a safe in a bank vault in another galaxy, and the people who know the combinations for the safes and bank vaults died ten million years ago. Still, the gun goes off and fires the bullet through your foot.

Scheme - You shoot yourself in the appendage which holds the gun with which you shoot yourself in the appendage which holds the gun with which you shoot yourself in the appendage which holds the gun with which you shoot yourself in the appendage which holds ... but none of the other appendages are aware of this happening.

sh,csh - You can't remember the syntax for anything, so you spend five hours reading man pages before giving up. You then shoot the computer and switch to C.

SmallTalk
1 - You spend so much time playing with the graphics and windowing system that your boss shoots you in the foot, takes away your workstation, and makes you develop in COBOL on a character terminal.

2 - You daydream repeatedly about shooting yourself in the foot.

SNOBOL - If you succeed, shoot yourself in the left foot. If you fail, shoot yourself in the right foot.

SQL - You cut your foot off, send it out to a service bureau and when it returns, it has a hole in it, but will no longer fit the attachment at the end of your leg.

Unix

1 - % ls foot.c foot.h foot.o toe.c toe.o
% rm * .o
rm: .o: No such file or directory
% ls
%

2 - % ls -a --color
man src MakeFile
% rm * .o
rm .o: No such file or directory
% make ShootFoot; make INSTALL

Visual Basic - You'll shoot yourself in the foot, but you'll have so much fun doing it that you won't care.

WorldBuilder - You can't shoot yourself in the foot, but you can shoot the feet of plenty of monsters.

Xbase - Shooting yourself is no problem. If you want to shoot yourself in the foot, you'll have to use Clipper.

XML - After describing the pistol, the bullets, and your foot in minute detail, you find out that you can't actually get to shoot your foot. So you begin describing your disappointment instead.

Z - You write out all the specifications of your foot, the bullet, the gun, and the relevant laws of physics, but all you can do is prove that you can shoot yourself in the foot.