How to Determine Which Programming Language You're Using

The proliferation of modern programming languages which seem to have stolen countless features from each other sometimes makes it difficult to remember which language you're using. This guide is offered as a public service to help programmers in such dilemmas.

What is possible with different languages when the specs say 'shoot yourself in the foot':

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".

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

Ada - 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."

Assembler - 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.

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

csh,etc. - 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.

FORTRAN - 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

PL/I - You consume all available system resources, including all the offline bullets. The DataProcessing & Payroll Department doubles its size, triples its budget, acquires four new mainframes, and drops the original one on your foot.

PL/I - you forget to declare foot; the compiler graciously goes with the default for lower anatomical feature; you run the program and shoot yourself in the *ss.

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

COBOL - USING 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 tied.

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...

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.

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

FORTH - Foot in yourself shoot.

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

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

APL - you spend a day writing the function to shoot yourself in the foot; this is followed by a week getting it down to a single line of code; then you show it to a co-worker only to find to your chagrin and embarassment that the strange symbol you had been wondering about all these years was the builtin operator used to......

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

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

SNOBOL - You grab your foot with your hand, then rewrite your hand to be a bullet. The act of shooting the original foot then changes your hand/bullet into yet another foot (a left foot).

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

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

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.

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

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

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

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

Smalltalk - 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.

Smalltalk - you shoot yourself in the foot, "Walkback: Foot does not understand: bullet"

Smalltalk - (Gun new target: Foot left) fire.

Smalltalk V/Win - aim at left foot, quickly port code to Smalltalk V/OS2 and left ear is hit.

PARTS - You spend so much time having fun designing a super cool UI for your app and integrating Cobol, C and SOM objects that your boss comes in and shoots you in the foot, telling you to develop in Smalltalk. He comes back the next day, shoots himself in the foot, goes back to his office and starts using PARTS.

PARTS - You want to shoot yourself in the foot, but can't pull the trigger without learning Smalltalk (see Smalltalk).

PARTS - You try to shoot yourself in the foot, but end up hanging yourself with a little green rope instead.

PARTS - You hook the slider to the dial, the dial to the gun, and the gun to the slider. You have so much fun building your demo, you forget about your foot.

PARTS - Open a workbench. Drop a LinkJunction. DirectEdit it to "Gun". Drop a ValueHolder. DirectEdit it to "Foot". Link *open* to Gun>>trigger. Link Gun>>triggered:"shot" to Foot>setValue:*. Done. Spend the rest of your day in the Icon Editor trying to draw a foot.

PARTSTalk - Write a script - Fire - for the Gun. At the end of the Script, write triggerEvent: #hasFired: arguments: (Array with: Bullet). Link #hasFired:* to Foot>setValue. Done. Come back a week later, show All Links. Wonder what this does!

PARTS and **Smalltalk** - Derive Foot from PARTSPrimitive. Create DLL.

Create Parts. Place on Workbench. Drop. If the Part is *really good* it will check for any guns already on the workbench, automatically link, and shoot itself!

PARTS and **VisualAge** - Both *your* feet are already held to the fire. If SOM is really the answer, sit down. BIND your feet. If you don't HAVE bindings - yet - Demand that Objects be truly transparent regardless of Vendor or Source. Aim at the Vendors feet, start shooting. Continue until your bindings arrive!!!!!

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

Visual Basic (again) - You'll really only APPEAR to shoot yourself in the foot, but you'll have so much fun doing it that you won't care.

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

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

Prolog - You ask Prolog if there is a bullet hole in your foot and it shoots you in the foot as a side effect and answers yes.

VisualAge - You borrow your best friend's gun, learn how to shoot, shoot your best friend in the foot, and announce to the world that you've invented the gun.

VisualAge - Forget it! You can't shoot anything. They didn't have guns back in Camelot. At best, you can follow a new standard and ride roughshod over your Friend's Foot and those of his followers.

Cross System Product - Spend a great deal of time defining OS independent Feet, Guns, and Bullets all in one environment. Generate. Load. Run. Shoot. Check your foot for results. Discover that you did shoot yourself in the foot - but it can only be fixed in the original environment.

HighPoint - Arm yourself with SOM and CSP. Announce your *intention* to shoot someone else in the foot.

CORBA - Define a Foot. You can now shoot it from ANY language. That's easy. Getting your Foot defined in the first place - that's hard.

CORBA with IIOP - Define a Foot. Export It. It can now be shot from any language, anywhere with one small restriction: the first shot costs $30K.

Workplace Shell - You correctly aim at your foot and discover the gun is in a different thread. Reach for the gun and your feet are cut off at the knees. You don't know how it happened - its private.

IBM OS/2 - SYS3175. The system has detected that you Shot Yourself in the Foot and are quickly bleeding to death. Please contact your Personal Service representative.

Microsoft WindowsNT - Wait!! Don't shoot yourself in the Foot *now* - just wait for Chicago. THEN you can load the gun. To actually shoot requires you be in Cairo.

Apple System 7 - Oh! Go ahead and shoot yourself in the Foot. We did.

Workplace OS - You may or may not be able to shoot yourself in the foot - it depends on your personality.

370 JCL - You send your foot down to MIS with a 400 page document explaining how you want it to be shot. Three years later, your foot comes back deep-fried.