Academic Programmers — A Spotter's Guide

Pete Fenelon

Department of Computer Science,
University of York

Introduction

During the course of a computer science research project (or even a DPhil) it is highly likely that a researcher will have to generate at least a couple of lines of code. Most researchers fall into a number of well-defined categories when it comes to programming. This handy guide for supervisors, other researchers, or the plain bored helps you to identify some of the prime suspects . . .

Disclaimer: this was written when I should have been concentrating on my current research project, the one my previous contract was for, *and* my DPhil thesis! No resemblance to individual researchers alive, dead, or at York is intended.

1. "I Am The Greatest"

Firmly believes that he is the greatest programmer to have walked the earth and has the three-line version of Tetris to prove it.

IATG spent most of his undergraduate days in the terminal room and only got a degree because he could break security and decrypt the exam answers. Thinks in a mixture of C and assembly language, thinks Real Programmers are sissies, has memorised even the unwritten volumes of Knuth (who he believes sold out the moment he started writing TEX) and has most of the source to obsolete Unix kernels in his room. Has VMS source on microfiche, mysteriously acquired. Knows what the Lions Book is and has his own n-th generation copy of it. Has played a Plan 9 distribution CD-ROM through an audio player for fun.

Nobody else can understand IATG's code, which suits him fine, and absolutely nobody can use his customised environment, which also suits him because it means he doesn't have to answer questions about it.

Absolutely lethal on any project which involves collaboration, documentation, theory, or distributing code to other sites; IATG is best steered away from research and into hacking for GCHQ or similar.

2. "Internet Vegetable"

Probably spent most of his early career sitting next to IATG in the terminal room but was reading the news instead. IV brings a new approach to research programming. IV has a near religious belief that the Internet is infinite in size and therefore must contain, accessible via anonymous FTP, precisely the package which is needed to solve the problem at hand. The problem is of course that it either won't run on any of the machines on site and necessitates wholesale upgrading of software and hardware, or requires "just one more" patch to be obtained via the net. When it does work, often IV is instantly disappointed by the vast shiny new package and throws it all away in favour of some other package which "may well do the job". IV knows where to get an infra-red weather map of Hawaii for 1963 and a program to display it on a TI 99/4A emulating a Commodore-64.

IV can survive at sites with tolerant sysadmins and good connectivity — but use of disk space is tremendous and demands for OS upgrades, net bandwidth, and new disks phenomenal . . . once in a while IV finds something useful but is usually too busy looking for something else to actually install it or port it.

3. "Rabid Prototyper"

RP has read all the books on software engineering and believes that you should build things incrementally and use prototypes. Unfortunately, RP takes it to extremes and re-starts from scratch almost every day, trying new approaches, new user interfaces, even new languages, in an attempt to achieve a design of such amazing elegance that all who see it shall be awe-struck. Unfortunately, every time RP has a new idea it means all the old work is thrown out, and in many cases this happens before any decent components are written.

RP tends to have an arcane knowledge of Unix tools like lex, yacc, Perl, and Awk, and RP systems are usually held together by the most incredible array of plumbing since an Italian hotel water closet.

With someone to catch the pieces thrown away by a good RP things might actually get done, and it can't be denied that they often have good ideas, but the sheer lack of commitment makes them impossible to cope with for any length of time.

4. "Get New Utilities"

GNU, as suggested by the name, believes that there is One True Source of good software and it's the Free Software Foundation. Not content with the perfectly good utilities and compilers shipped with the system, GNU has to have gnu-cat, gnu-rm, gnu-everything before any work can be done. Of course, because gnu-anything usually requires gnu-everything-else to build, you always end up with a complete set of gnutilities filling your user disk leaving no space for research work. Come to think of it, because GNU is always applying patch 9.4.32.4.4.12 to the gnuseless programs he needs to build more gnuseless programs, there isn't any time either.

On the rare occasions when GNU actually does write some code it'll require the entire GNU CD-ROM to be shipped with it before it'll even compile on a standard machine.

Useful to have at least one of them around, but beware getting two GNUs, since they'll inevitably both want their own collection of software . . .

5. "Square Peg; Round Hole"

SPRH wrote a good program a couple of years ago, which solved a problem nicely and had some useful bits in it. Since then, however, SPRH has moved to a new project with new objectives. This doesn't matter, since as far as SPRH is concerned *all* software is reusable.

The old program will either grow enormously into a multi-modal, immense crock held together by hidden parameters, mode bits, recondite options, and obscure data types, or will shatter into a disk full of libraries, macros, class hierarchies, and fiddly little separate programs which used to fit together but now need so many intermediaries to communicate that they've become incomprehensible.

SPRH often looks productive, but that's because most of the work was charged to another project five years ago, or whatever work is going on is just another attempt to bludgeon code into an alien Weltanschauung.

6. "Objectionably Oriented"

OO experienced a Road To Damascus situation the moment objects first crossed her mind. From that moment on everything in her life became object oriented and the project never looked back. Or forwards.

Instead, it kept sending messages to itself asking it what direction it was facing in and would it mind having a look around and send me a message telling me what was there . . .

OO thinks in Smalltalk and talks to you in Eiffel or Modula-3; unfortunately she's filled the disk with the compilers for them and instead of getting any real work done she's busy writing papers on holes in the type systems and, like all OOs, is designing her own perfect language.

The most dangerous OOs are OODB hackers; they inevitably demand a powerful workstation with local disk onto which they'll put a couple of hundred megabytes of unstructured, incoherent pointers all of which point to the number 42; any attempt to read or write it usually results in the network being down for a week at least.

7. "My Favourite Toy Language"

MFTL knows the solution to the problem. The only problem is, we haven't got a compiler for the language that it should be implemented in. MFTL knows only two languages; his favourite toy language and the language you need to compile its compiler. (If a language can compile its own compiler then it isn't a toy!)

The problem with toy language compilers is that the code they generate is often inefficient and impossible to debug; however good MFTL is as a programmer the system will be huge and clunky . . . in many cases the toy language also needs extensive runtime libraries and support tools to be distributed.

Is more likely to spend time tinkering with the toy language compiler than actually working on the project; dreams of the day when the toy language is implemented in the toy language, and will probably resign as soon as it is, unless it's a Functional Programming project — almost all of them are about writing compilers for someone else's toy language.

8. "Give Us The Tools . . . "

GUTT already has the software to solve the problem. Whether custom-written or commercial, it's excellent stuff and works nicely; it's robust, it's simple and neat. It often originated from the last site that GUTT was employed at and there's the problem . . .

It doesn't run on any of our machines. GUTT seems to have been living in an alternate reality in which Scrotely Whizzbangs running ScroteOS and StainedGlassWindows are the most popular computing environment and has begged, stolen, borrowed, or even written software to suit.

The problem is of course that outside Ruritania nobody on Earth has ever heard of Scrotely Systems and the software isn't worth a row of beans to anyone . . .

Since Scrotely went out of business five years ago, truly great GUTT people spend months trying to write a Scrotely emulator on your local machines; mere mortals spend their time posting to comp.sys.scrotely and comp.sys.foobar to ask whether anyone has ever tried porting anything to a Foobar 250 . . .

9. "Macro Magician"

Macro Magician believes that programming is obsolete because you can make any program sit up and beg via use of its command or macro language; MM can solve your problem with a quick macro here and a bit of shell script there to hold it all together. There are two types of MM: the Unix Macro Magician (UMM) and Micro Macro Magician (MMM).

Whether it's solving the Towers of Hanoi in vi or sorting lists in TECO, UMM knows how to do it. UMM pipes his .profile through the C pre-processor and watches it rewrite itself every time he logs in; the vast majority of UMM systems are implemented in Emacs Lisp and require all 2.5 Gbytes of the latest distribution before they'll even think about running. They usually take at least as long to run as to write.

At the other end of the scale, MMM is into HyperCard, ToolBook, and other BiCapitalised pieces of syntactic sugar, although also relishes the chance to delve into the macro languages of word processors, databases, and spreadsheets, preferably all at the same time; ideally using everything to build an application which takes a week to start up and keeps flashing up obscure menus and dialogue boxes.

No cure for either of these, sadly. Best bet is 240 V through their chair.

10. "Nightmare Networker"

NN relishes complexity. The database runs on an IBM somewhere in Canada; the X-windows front end on a Hewlett-Packard in Switzerland, the inference engine on a Cray in Indonesia, and the program itself on Voyager II . . . each part of the packages employs different comms protocols depending upon a wide range of factors including the phase of the moon . . .

There is no doubt that NN can create a system which works, but it's impossible to explain to mere mortals and keeps getting more and more complex. NN firmly believes that "it's all virtual anyway", unfortunately including such things as execution time and network bandwidth.

NN can be exhilarating to work with but also infuriating — never let NN tinker with your workstation because in no time flat it'll be running EBCDIC to SixBit translation software routing X.500 address requests from Uzbekistan to Ouagadoudou via a steam powered TCP/IP to Alohanet gateway in Auchtermuchty . . . Best relegated to a support job if at all possible.

11. “Configuration Unmanageable”

Never produces anything remotely useful, but has all the crud that he has ever written under a wonderful change-management system. Literally everything he’s ever written, from “fank you leter to aunty doris by me age(6)” to his underpants, is stored under RCS with proper versioning etc. Want version 8.2 of his O-level English essay? There it is.

Somewhere in there are 14000 versions of the source to the current project; CU saves and generates a new build every time a single character changes because “you can never be too sure” . . . CU is also an archive freak and his office is habitually filled with magtapes, QICs, and Exabytes containing a complete backup of the revision notes about the versioning policy of the document identification scheme for the change-management procedure for the backup procedures for the system.

Words like “anal retentive” have been used to describe CU but he can’t look them up because there’s no longer enough space for the online dictionary . . .

Impossible to work with and to get any work out of. Is more likely to be out spotting trains or collecting stamps than working in any case.

12. “Artificial Stupidity”

Two types of AS researcher exist and both of them are hell to live with. The more traditional type spends most of the day counting parentheses in epic Lisp programs and trying to tell Prolog systems about families. If he gets mixed up he just fires up Eliza and tells it about his family until it crashes with boredom . . . Truly great AS researchers get their Prolog programs to talk to Eliza about their families and spend the rest of the time at conferences.

The new type is into Neural Networks and spends hours (and megabytes) with kludgy, flaky software creating arrays full of zeros and the odd one here and there for good effect. Interminable programs generate huge files with these, in an attempt to prove that you can tell the difference between margarine and butter in less than ten hours . . . Occasionally has video cameras and image processing software — run like the clappers when this happens because invariably it will be unable to distinguish you from a picture of Cecil Parkinson or suchlike.

The problem with AS researchers is that the systems they create are at least as stupid as the people who create them. Avoid at all costs.

13. “Number Crusher”

NC knows the solution to the problem — it’s a couple of seventeenth-order non-stiff bloody hard integral equations, and there’s a routine somewhere in the NAG library to solve them. Isn’t there? Unfortunately NC isn’t much of a programmer (strictly FORTRAN or the most K&R-ish C you’ve seen for years) and isn’t quite sure which routine, or which parameters, or for that matter which library . . .

NC is often not a computer scientist — physics or maths backgrounds are common — and tend to have the clunkiest working environments on machines you’ve ever seen. Keep all their files in one directory and name them F44433232.DAT and suchlike. Almost always have a

Julia set as their screen wallpaper (Mandelbrot is a bit old hat and doesn't take up enough processing power . . .).

Knows a lot about the likes of Uniras, GINO-F, and similar. Can be relied upon to have the floating point format for the machine tattoed inside her eyelids and mumbles Denormal, Abnormal, Inf, NaN! in her sleep (while the system recompiles). Is the only person in the office who can remember O-level maths and as such is occasionally useful.

14. "Meta Problem Solver"

Believes that the problem can be solved by either finding a solution to a well-known problem which can map onto your problem, or by creating a tool which can generate a program to solve the problem. MPS usually knows a lot about automata, language theory, and obscure algorithms and revels in complexity.

Often sounds plausible, but the meta-problem which MPS keeps trying to solve generally generates a whole slew of meta-meta-problems, and the meta-meta-problems in turn infect the project with meta-meta-meta problems, and eventually MPS either disappears up his own rear end or ends up having to solve the Halting Problem before he can get anything else done.

An MPS on the team can be extremely exhilarating, but most of the time it's downright difficult. Many MPSs were formalists or mathematicians in another incarnation, which can make them difficult to deal with. Their programs often run better on a whiteboard than on a computer.

15. "What's a Core File?"

The deadliest of the deadly, WACF drifted into the Department from some other planet and still believes that computers are magical, strange, contrary beasts. Every login session is a strange and terrible adventure. Has a filespace full of .dvi files, editor backups, text files called aaaa, bbbb, cccc, xxxx and suchlike, and a few core dumps (usually caused by the window manager or kernel, since WACF rarely programs). Generally uses one or two killer applications which hammer the fileserver or the net, but forgets to kill them off and ends up with seventeen text editors, eight window managers, and a dozen or so clocks running at any one time.

As long as you can convince WACF not to do any programming you might have a chance of getting something done. Ideally one should buy them a PC or Macintosh which isn't attached to the net. Oh, and protect the system files, because WACF has been known to delete things like MSDOS.SYS to save space.

16. "I Come From Ruritania"

Used to be the best programmer in Ruritania, where computers run on steam and use trinary deontic logic with lots of don't cares. Regards 8k of memory as a paradise of unheard-of proportions and doesn't trust windowing systems. Speaks fluent Ruritanian and starts off seeming to speak good English, but gets confused whenever the phone rings so doesn't bother answering it, only believes things other Ruritanians tell him, and insists on using the office as an informal Ruritanian social club.

Some ICFRs are actually excellent programmers by any standards, but the effectiveness of their work is blunted rather by the fact that if you can persuade them to write user documentation it will display a choice of grammar and vocabulary which is at best idiosyncratic and at worst somewhat like a Sun manual; added to this the code is of course all commented in perfect Ruritanian. It's often fun to dig out their CVs or read their mbox files, which they often seem to leave unprotected. Unfortunately in several cases ICFRs have left their girlfriends back home unprotected just before coming to the UK; being present at the birth by email is a difficult option.

17. "Old Fart At Play"

Every institution has one. OFAP has been around since (as a bare minimum) the mid-sixties and regards such arriviste architectures as VAX as being unproven and too modern. OFAP regards the PDP-6/10/Decsystem-20 line as being the One True Architecture and reckons characters are six bits wide, never mind all this ASCII rubbish, let alone Unicode. Delights in explaining the CAILE instruction at coffee breaks and maintains an FTP archive of old PDP-10 operating systems. Was mentioned in HAKMEM and is delighted when he finds anyone else who's heard of it.

OFAP has occasionally been convinced to port some of his code to Unix, but of course never got further than V7. Once tried to port Spacewar to a modern machine but it wasn't fast enough. Knows that Sketchpad is the greatest drawing program ever. Knows what all the funny mode bits in obscure TECO Q registers are used for, and exploits them in his programs, which are an unholy mixture of assembly language and TECO macros. Dangerous, usually has a beard (even if female), but is useful to have around because s/he has seen it / done it all before and knows the tricks — just don't let OFAP implement anything.

18. "I Can Do That"

ICDT tends to be an enthusiastic new graduate and mistakes user interface for functionality. That is, once ICDT has seen a program running he believes that he can "knock up a quick hack to do that in a week". Four or five years later the "quick hack" is still unfinished, because ICDT doesn't understand the underlying semantics or data structures.

Combining an ICDT with another programmer is often a damn good idea as long as someone can curb his enthusiasm. There is a slight downside in that most ICDT programs are predicated upon a huge and unreliable user interface class library — InterViews is particularly popular for creating mock-ups of programs that will never work, although in this enlightened day and age Visual Basic and Visual C++ are starting to take over as media for creative delusions.

May be a useful member of an HCI group or some other motley crew in which programming skill isn't important but getting pretty pixels on screen is vital.

19. "What Colour Should That Be?"

Has read all of the books on HCI and believes all the contradictory stuff that's contained in them. Always has a more expensive machine than you, usually with a very nice colour screen, sound card, Dataglove, voice recognition equipment, etc. — and no keyboard, because

WCSTB can't (won't?) type. Is more likely to be a psychologist or sociologist than a real computer scientist.

WCSTB basically prefers tinkering with typefaces, colours, screen layouts, and window-management policies to programming, although most WCSTBs have a working knowledge of some of the surprisingly grubby depths of either X or Windows, in order to facilitate the above. A typical WCSTB "Hello World" program is four hundred lines long and takes up a meg and a half when linked, but is essentially a complete multimedia experience with a non-threatening user interface and configurable options; at that rate it's perhaps surprising that none of the WCSTBs ever get anything more substantial written.

20. "It's Safety Critical . . . "

ISC is the barrack-room lawyer of the research community. Since the application areas in which he works are closely allied to blowing things up / stopping things from blowing up he takes a considered and principled approach to software development for safety critical systems — his claim is that "all software is unsafe and I'm damned if I'm writing any to complicate the issue".

In theory this is fine, but occasionally ISC is forced into writing some code by whoever holds his grant. Depending on what sort of safety critical project he's involved in, this will either be low-level bit-twiddling in C, PL/M, or assembler on a single-board computer (which ISC secretly loves because you basically don't have to do any V&V on it) or will involve interacting with twenty different CASE tools, eight design notations, and four formal methods with subtly incompatible semantics. Tends to be employed on long contracts, and with a development process like that can you blame him?