WORKING NOTE AC68

Alan Creak
6 March 1998

WHY DO COMPUTISTS LIKE GIBBERISH ?

(and related topics)

I take as my text this excerpt[1] :

> *These hints are intended for those using the Fortran compile, link, and go routines. Many problems are caused by inappropriate file allocation. First of all, ensure that your datasets reflect the size of your input or output data. Secondly, if in doubt about record formats, use RECFM V instead of FB or VB. Some people use VBS (Variable Blocked Spanned).*
>
> *The maximum figure for block size (BLKSIZE) is 32760. The block size sets a limit for the record length (LRECL) as follows. If the record format (RECFM) is F or U then LRECL may not exceed BLKSIZE. If RECFM is D or V then LRECL may not exceed BLKSIZE-4. However, if RECFM=VS then LRECL may exceed BLKSIZE.*

I chose this item not because it is an extreme example of the phenomenon I shall discuss, but simply because it happened to come to hand at a convenient time.

If you have hastened to check the reference, you might be surprised to find that a 1989 document should come to hand at a convenient time in 1998. Your surprise would be justified; the "convenient time" mentioned was in 1989, and it was then that I first vented my frustration by beginning this note. That I have only recently brought it to an approximation to completion (I expect never to consider it finished) is no indication of an abatement of the phenomenon, and over the intervening period I have continually, verging on continuously, been irritated by examples.

My complaint is simple. It is that, given half a chance, computists seem to delight in finding the most obscure possible way of saying anything. Other people (notably, it seems, mathematicians) attain marvellous heights (or depths) of obscurity in programming languages[2] , but we do it in English. Is this really the best we can do after forty or more years of development ? Accepting that we do need some technical vocabulary, shouldn't we be able by now to talk about computers in terms accessible to ordinary humans ?

Perhaps we should, but we don't. I was bewailing that fact to a friend some time ago, and pointed out that "monitor" was used in at least two different senses. She thought for a moment. "So it is !", she said in surprise, and listed the two meanings - one of which I hadn't thought of. (For reference : monitor system, synchronising monitor, and monitor meaning screen. You can add performance monitor if you like.) If anything, it gets worse; the current wave of inanity covers the internet, Java, middleware (hardly self-explanatory terms !), and other developments which are inoffensive in themselves but become submerged in jargon as by a process of spontaneous generation.

Some of us care. Katzenberg and Piela[3] , for example, discuss the problems of selecting good names for use with computer systems, where good names are defined as those which are easy to learn and remember accurately. As an interesting sidelight, they also quote other research which has shown that if you're looking for a word to use as an instruction that means "delete", and you can choose from the set "delete", "ragole", "parole", "dlt", and "fnm", then the best choice is "delete". That, at least, is reassuring. But it isn't as funny as it sounds; people who use "mv" to mean "rename" should take note.

All things being considered, though, perhaps we don't care very much. I present a lecture course on operating systems, and a part of our routine is to request students to assess their courses at some time during the year. One assessment of my course, from a - perhaps fortunately - anonymous student :

> *Good if you are an arts student & good at waffle & putting together words.*
> *Not good for a science student.*

On the whole, though, I don't think that attitude is typical. The computists I know are not by nature aggressively illiterate people - indeed, some of my best friends are computists, many are highly articulate, and though not all are conversant with a wide range of English literature none affects to despise it. Neither are they, like some mediæval alchemists, concerned to protect the mysteries of their craft. On the contrary; lots of them are all too eager to tell you about it.

They are honestly surprised that people don't understand.

SHALL I COMPARE THEE TO AN ENGINEER ?

Engineers also find themselves in the position of needing to explain technical matters to people with comparatively shallow technical backgrounds. How do they fare ? I suspect they do much better - but they usually have some sort of everyday model which they can use to illustrate their material. Civil and mechanical engineers deal in the main with objects of common experience, and need no model. Electrical engineers can get quite a long way by using a hydraulic model : high voltage equals high water pressure, high current equals rapid flow, and so on. Of course, the model isn't perfect, but it helps.

Scientists too have their models, but they have contrived to pull off a remarkable confidence trick : their models are more abstract than the phenomena which they represent, but people accept them nevertheless. They happily put up with "explanations" in which simple observations are interpreted in terms of very obscure ideas. ("Why did the stone fall when I let go of it ?"; "Because of gravity.". Think about it, and contemplate relativity if you dare.) Somehow it works. It seems that any model at all will do.

What sort of model has the computist ? None. We deal with ideas which have no ready counterpart in most people's everyday lives. You can tell people *what to do* to drive a programme, but it can be much more difficult to explain *how it works.* (Confusion between these two notions is not restricted to computists; I have asked how a photocopier works, only to be presented with operating instructions.) All in all, there is little comprehension outside the computing milieu of what computing involves, so our students are still asked if they spend all their time writing programmes. Consider this comment from a graduate student some years ago[4] :

> *"I have many times come across the attitude that Computer Science is merely teaching programming, and nothing else(*). Some people have thought that a masters degree in computer science is nothing but writing a big programme, and some cannot understand what I am doing if I do not have a programme.*
>
> .....
>
> *"(*) 'Do you mean to say you have to know how the *****s work!!' - A biochemist examining the piece of paper I was using as a dart."*
>
> (My string of five asterisks.)

Johnson[5] suggests that it is this lack of a model which drives computists to liberal, not to say obsessive, use of metaphor in their language, and discusses the phenomenon with some sympathy. I don't mind the metaphor so much, but I wish that we could choose our metaphors with some consideration for what they mean, then define them, then use them consistently.

The trouble runs deep and starts early; the very name of the subject is an example of the problem. Is it really sensible to speak of computer *science* ? I don't think it is[6] ; I would prefer to regard the topic as

an engineering discipline. That leaves me in the anomalous position of working in a department devoted to a non-existent subject. (I did urge, before the department was established, that it should be called the Department of Computing, but my urges fell on deaf ears. Unfortunately, I can't prove this assertion, as it was a long time ago, and documentary evidence seems to have vanished.) I am consoled to some degree by the empirical observation that the salary seems real enough.

THINK (AS THEY SAY - OR SAID - AT IBM).

Programmers have peculiar minds; they have to be able to think on several levels at once, they revel in puzzles with ingenious and non-obvious solutions. That's why they're fascinated by "one-liners" - programmes which do amazing things, but are expressible in a single line of text. The language APL was perfectly adapted to this purpose; modern programmers have to make do with C, which is serviceable but not inspired. (It must be emphasised that *good* programmers are also very well aware of the defects of that programming style, and know when to discipline themselves to write well structured code. But the fascination remains. From another graduate student's assignment report[7] : "I feel that this is a very *elegant hack*". And for a definitive example, notable for its literary excellence, see the comment by Ed Nather[8] on "Real programmers write in FORTRAN".) Acronyms abound not because they are thought to be impressive, but because they're clever and perhaps funny - it's a game. There is no intention to obfuscate, but if you have to have a name for something, why not have an amusing one ? If nothing else, it's easier to remember.

(Acronyms which are unpronounceable, unwitty, or just uninteresting in any way excite derision rather than admiration - one of the reasons, perhaps, why, THINK or not, IBM is regarded so negatively. It just doesn't impress them as clever or entertaining.)

Good programmers are programmers because it's exciting. They are not doing a job; they are working out a vocation. They want to do exciting things, and they are impatient with anything - such as routine work, or the need for documentation - which gets in the way. They see programming in much the same way as some others see pop music, and one can draw many analogies between the two obsessions. They are both closed communities, more or less incomprehensible to outsiders; they both change with bewildering speed, and their practitioners therefore need (or believe they need) new terminology very frequently. The new jargon is not really intended to be arcane, but no one minds if it becomes so, for everyone of importance will understand it anyway. There is no attempt to "spread the word", though anyone who can join in the rites is welcomed. If you can't understand the rites, no one will stop to help you; you are an outsider.

It's a closed world; why would you want to go outside ?

Except, of course, to get a job. Jobs are an unfortunate necessity, and there are hardly any jobs offering *real* programming. For the rest of us, this is just as well; if it were not so, we would quite probably not have even the often meagre and incomprehensible documentation on which we now barely survive - or, perhaps, don't. This view of employment can encourage its adherents to lump everything connected with the job, except programming, into the same despised category. A job might mean wearing a tie, which is admittedly pretty ridiculous; a job might mean writing documentation, which is not. Unfortunately, there is a tendency to see both of these as arbitrary and capricious constraints.

I SAY, I SAY, I SAY ...

The vocabulary of computing is perhaps unique in the philology of science and technology in that it was to a great extent not designed by the people who developed and understood the technology, but by salesmen and advertising agents - and, interestingly, in part by the real programmers, who developed but often didn't understand the technology, reinventing wheels in abundance and giving them different names. Johnson[5] remarks : "... this is admittedly the language of sales and marketing and should perhaps not be criticized too severely". I suggest that, on the contrary, it should be criticised with vigour and zeal. What

other discipline meekly allows people with no obvious claim to expertise (and sometimes very obvious absence of expertise) to construct - and, worse, to alter - its vocabulary ?

For example, look what happened to stacks. For a long time, we've known what a stack is. It's a data structure which was originally called a push-down or pop-up stack, typically used to remember items while some other activity went on and to recall them in a proper "bracketed" order. It's had other connotations in its time : a *Dictionary of Computers*[9] (which, while we're in critical mood, could perhaps better be entitled *A Dictionary of Computing*, but we'll let that pass) defines it as a sort of buffer without specifying any structure, but it was certainly used in the push-down sense at around the same time[10] . The CP component of IBM's VM operating system had a structure called a *console stack*[11] , which combined the attributes of stack and queue. But then the infinite - or, at least, inscrutable - wisdom of the Apple Corporation decreed that a stack was a circular list[12] , and did so in such a way (by attaching the usage to their Hypercard product, specifically designed for use by non-specialists) as to guarantee that confusion will reign.

One might bewail some of the excesses of the more traditional scientific terminology, but at least it is usually precise. Computing terminology is frequently chosen for its impact rather than its precision; the consequence is that there might be several ways of describing a phenomenon, and each description is sufficiently vague to include several other things too. Just what is a fourth-generation language ? What is object-oriented programming ? What do on-line and off-line mean ? We don't improve the situation when we forget what we're doing :

NAME	*lpr - off line print*
SYNTAX	*lpr [option...] [file...]*
DESCRIPTION	*The lpr command uses a spooling daemon to print the named files when facilities become available.*

That's taken from the description of the system instruction lpr(1) in the Unix manual, as implemented under Ultrix. It's reasonably clear, once you're used to the conventions of the manual - though one might wonder why it can't just say "The lpr command is used to print the named files". But "spool" is an acronym - quite a good one, for IBM - for "Simultaneous Peripheral Operation *On* Line", which makes it an odd choice of word for an operation described as "*off* line print".

My view, for what it's worth, is that the description is wrong : lpr prints files on-line. I am supported in this view by my dictionary, where we find this definition :

> ***on-line*** *A part of a computer system is on-line if it is directly under the control of the central processor.*

The same dictionary, incidentally, offers a marvellous example of the phenomenon we are discussing, which demonstrates that the problem is by no means new :

> ***foreground processing*** *1. In a* multi-access system, *processing which is making use of* on-line *facilities. 2. High priority processing which takes precedence (as a result of* interrupts *) over* background processing. *3. Low priority processing over which background processing takes precedence.*
>
> *As definitions 2 and 3 are directly contradictory and definition 1 has a related but different meaning, this phrase should be used with caution.*

An interesting inversion of meaning can be seen in the use of *synchronous* and *asynchronous*. My nearest English dictionary[13] defines "synchronism" as "Concurrence of two or more events in time, coincidence, simultaneousness ...". Now consider this passage[14] :

> *A* synchronous *operation invocation causes the thread to migrate to the object executing the corresponding method. When the method completes, the thread migrates back to the object that invoked the operation. An* asynchronous *invocation, on the other hand, causes a new thread to be created. No order is assumed between the events of the original thread and those of the created thread.*

- so in a synchronous operation, the two threads certainly do not operate at the same time, while in an asynchronous operation they may. It's easy enough to see how the inversion of meaning took place - but why didn't someone, somewhere along the way, see how silly it is, and do something about it ? It's fair to remark that one should make allowances for "synchronized", because it's had a disturbed childhood; it had to endure SYNCHRONIZED LEFT and SYNCHRONIZED RIGHT in Cobol, where time isn't involved at all.

In another mildly intriguing use of metaphor, we no longer run programmes - we *launch applications*. (Or, rather, you might - I still run programmes.) My faithful computing dictionary again :

> ***application*** *The particular kind of problem to which data processing techniques are applied ...*

- so, for example, accounting, matrix arithmetic, and computer graphics might be seen as computer applications. But, though a programme might certainly be a "kind of problem", I'm sure that's not what the dictionary meant. A programme which addresses an application was called an *application programme*, which is reasonable enough; to call it an "application" seems, once again, to add an unnecessary element of confusion. The linguistic transformation seen in this shift of terminology is interesting; the compound term is abbreviated by omitting the more significant part. The same transformation operates in the contraction of "run-time environment" (clumsy, but comprehensible) to "run-time".

Incidentally, to judge by the number of approximate synonyms which one finds, the word "programme" itself seems to be something of an embarrassment. (Perhaps that has something to do with the spelling : see below.) Even leaving aside applications (or, worse, apps), there are routines, codes, packages, systems, products, software.

But, whatever we call it, why "launch" it ? The term might have been used earlier, but I first came across it in the context of the Macintosh system. Is it part of the desktop metaphor, perhaps ? The only time I've come close to launching anything on my desktop was when a mug full of coffee came away from its handle while I was holding it over my desk. It is not a part of the desktop metaphor which I could recommend to anyone.

I could go on. (I frequently do.) There is no dearth of examples, but I'll give just one more. I choose it because it has always intrigued me. It is the common use of the term "spell checker". Now, we all use spelling. Some of us are supremely, arrogantly confident that our spelling needs no checking, and that in cases of dispute we are more likely to be correct than someone else's software; others value software assistance in getting the spelling right; a minority sadly believe that after having their spelling checked by software the result must be correct. But how many of us use *spells* ? We have heard of Programmer's Apprentices[15] - but surely a spell checker is a Sorcerer's Apprentice[16] , which so far as I know is still an opera rather than a practical software project. (Incidentally, for a parable on the dangers of imprecise use of language which also involves spells, I recommend *The truth about Pyecraft*, a short story by H.G. Wells[17] .)

- unless, of course, it's really a Sorcerer's App. Now, there's a thought to conjure with, and it has a fine dramatic sound to it. Speaking of drama ...

TALES OF DERRING DO.

A curious feature of the language of computing is its tendency to lean towards hyperbole, if not violence. Simple and commonplace operations are presented in heroic terms. You do not instruct a computer, tell it what to do, or direct it; you issue *commands*, a term carrying a strong sense of the exercise of a dominant personality. I could, in principle if not in practice, command a battalion; I do not command my computer, and to pretend to do so is as ridiculous as claiming to command a bicycle. Computer systems do not fail; they *crash*. The term seems fair enough when used in the context of a disc drive head, but in many other contexts would be more than a little incongruous were it not for regular usage. You do not make a file; you *create* it : you are a god. (There's a real example in the synchrony quotation above; did you notice it ?) A programme in execution is not stopped; it is *aborted*. Even the simple oblique stroke or solidus becomes a *slash*. Perhaps it is in this vein that a programme might be *launched* to be its author's fragile barque on the storm-tossed ocean of programming chance and fortune. Oh, to be a buccaneer !

Why all this posturing ? I don't think it's a need to add drama to an otherwise dull existence. Life as a programmer is often quite dramatic enough without embellishment. Neither is it that programmers are intrinsically violent people. Perhaps it is a natural consequence of a wish to describe the exciting events which do happen in what seems to be suitably picturesque language.

It's interesting to record an observation which might be seen as a counterexample. When Auckland University first acquired its IBM4341 system, running CP and CMS, there were not infrequent momentary failures of the operating system : the system would stop, then automatically restart itself within a few seconds. Following the lead of the IBM systems engineers, our computer operators began to use the phrase "the system *bounced*". This is surely a matter of giving a dog a good name in the hope that it will therefore not be hanged ! A bounce is a friendly, playful thing : it is not a word to associate with an event which could in some circumstances wipe out hours of work. I remarked on this observation to our head operator, who then confirmed my suspicions : he had accepted that a bounce was a comparatively innocuous event, without really considering its effect on people who were working at the time. My propaganda bore some fruit - the system didn't stop "bouncing", but at least the operators began to say, "the system failed".

An alternative view is that programmers really do believe that they are closely related to the Knights of the Round Table[18] . From my own experience, I would have rejected this supposition as absurd, but reflect on the Microsoft Corporation's view of an operating system[19] :

> *The IBM Personal Computer Disk Operating System (DOS) controls the movement of information in your computer. You can think of DOS as the wild animal trainer in the circus making the animals do feats of physical skill and daring. In much the same way, DOS controls the way your computer uses programs, games, and applications.*

If this is the corporate view of a microcomputer operating system, one can only speculate on how they feel about mainframes. (Perhaps it's only the corporate view; here's another view from a textbook[20] at about the same time :

> *An operating system is much like a traffic cop, directing the flow of routines and processes.*

The picturesque imagery remains, but the level of spectacle is rather more moderate. Or so one would hope.)

Where does all this violence come from ? Who is the enemy ? Is there an enemy ? Is it the computer ?

LOOK OUT, LOOK OUT, THERE'S A USER ABOUT !

No - the computer is, by definition, a friend. The enemy is anyone who might come between you and your computer by making you waste valuable programming time on squalid tasks which merely pander to those too idle or lazy to train themselves up to your level of understanding. The enemies are people who want documentation; they are people who want reports; they are people who want planning meetings; they are, above all, users.

A well tried psychological tactic in warfare is to find some non-human term with which to describe the enemy - the huns or the boches of the second world war, the commies and reds of the cold war, the gooks of Vietnam. The dehumanising term in the computer war is the *user*.

In this matter, though, the computer war is just a shade different from the more traditional sort, in that it's quite hard to think of any other snappy term which adequately describes people who use computers. In specialised circumstances, you can speak of clients or customers, but to do so demeans a relationship based on honest work to the level of a commercial transaction. Apart from that, other than rather clumsy circumlocutions about "those who use this system" or "people sitting at terminals", "user" is all there is. So surely sheer pressure of usage must have purged it of any dehumanising overtones it might once have borne ? Why do I inveigh so strongly against it ?

Because once I was on the other side of the fence. I worked in computer centres in their heyday, and well remember phrases like "it's only for the users"; "it'll do for the users"; "we don't need to tell the users". Perhaps it doesn't happen any more. Perhaps ? M.C. Board[21] , writing in 1991 :

> *Note the distinction made in the computer press between "users" and "tech types". Being a "user" suggests a degree of helplessness; being a "tech type" suggests some serious personality defects.*
>
> *"Power Users" obviously have inflated egos. There is no easy word for the middle ground.*

(And there's the hyperbole again, in "Power User". What a silly name ! It's almost as bad as Turbo-this and Turbo-that, which fortunately seem to have faded away. There at least I struck a blow for sanity; I ruled out the purchase of one computer because it was claimed to operate in normal and "turbo" modes. It is true that my protest had little effect, if only because I was not planning to buy a new computer just then, but the intention was honourable.)

Once again, the dismissive attitude to users wasn't deliberate nastiness, and Lepape and Talbot[22] have described the same phenomenon rather more charitably. To some extent, it was simply experience; the users were obviously quite ready to put up with a quality of service far below what was possible, so why should anyone sweat unduly to improve their lot ? I don't suppose many computists really thought too much about just why the users were so docile; it was because the computists didn't tell them what they could have if they really wanted it, and there was no real pressure on the computists to tell them. That might not be a particularly friendly attitude, but it was usual, and there's plenty of precedent for not going out of your way to make work for yourself.

It is rather sad, by the way, that this docility is still rife. In what other area of enterprise would people accept "guarantees" which are not ? Consider this fairly typical specimen :

> *The software and accompanying written materials (including instructions for use) are provided "as is" without warranty of any kind. Further, ****** does not warrant, guarantee or make any representations regarding the use or the results of the use, of the software or written materials in terms of correctness, accuracy, reliability, currentness or otherwise. The entire risk as to the results and performance of the software is assumed by you. ....*
>
> (My asterisks again.)

That came from the "Users manual" (surely instructions which tell you how to operate the Users ?) for a particular piece of software with which I am having a lot of trouble at the moment, which I suppose is why they put it there. I won't identify it, because closely similar admissions of incompetence appear on such a wide range of software products, and I don't see why any should be let off. The only redeeming feature is that the disclaimer of "correctness, accuracy, reliability, currentness" applies to the text itself - which is just as well, as you will discover if you try to work out what the third comma in the second sentence is supposed to mean. Perhaps it's the apostrophe from the manual title, dead and abandoned. But I digress.

To be fair, the computists don't usually go to the extent of engaging users in formal battles. They would be surprised at my perception of violence and enemies; perhaps their hyperbole has rubbed off a little onto me through long association. Their preferred policy is one of benign neglect - an attitude more feudal than military.

THINGS AREN'T WHAT THEY USED TO BE.

In the early days of computing, it was - reasonably - assumed that anyone near a computer would be an expert, probably in some advanced field of mathematics or physics which would now by no means be regarded as an appropriate qualification for an expert computist, but no matter. As time went by and computing grew into a semblance of respectability as a trade in its own right, a sort of priestly class of programmers and, sometimes, operators assumed the experts' mantle and tended the shrine, keeping the common people - the users again - at a safe distance. The experts were still alive and well, and ready, in principle, to mediate for the laity if the *deus in machina* should become angry with them.

This is no longer the case, particularly where microcomputers are concerned, but the memory lingers on. Even microprocessors have a questionable ancestry : the first people to use microprocessors as computers rather than as components of other systems were just the sort of programmers whom I have described, so the first microprocessor software was steeped in hackery. Nowadays, though, the old attitudes to the "users" are no longer appropriate. The experts aren't on hand any more - either because the people who do the work with computers use independent microprocessors, or because they use terminals or workstations separated from large machines by many layers of software, networks, communications lines, and so on. Now it is important that these people should understand computing in a way which was, perhaps, unnecessary during the reign of the experts.

But computing seems to have a way of growing more complicated. Some are unhappy about this, as reported of the early enthusiasts for Smalltalk[23] :

> *Alan Kay, speaking at a conference in Germany in 1985, referred to Smalltalk as 'an example of a once good idea that has gone way, way too far'. He should know if anyone, for it was he, and his learning research group at Xerox Palo Alto Research Center, that created the language in the early 1970s.*
>
> *The very name Smalltalk was a way of emphasising the accessibility of the language. 'It used to be small and usable by children, but it isn't now', said Kay.*

It isn't only Smalltalk. Lisp grew from a tiny and elegant functional language into a baroque monster which you can use for writing operating systems. Prolog regressed from something rather like first-order logic to a sort of badly done Fortran. Alternatively, sprawling languages can be developed by a much more efficient route in which the elegant stage is omitted entirely - think of PL/I, Ada, C++.

Others delight in "extended functionality", a related disease which attacks software rather than languages. Neat little software packages sprout more and more bizarre features, the screens of successive versions more and more crowded with incomprehensible icons. I am frequently urged, in defence of these GUI graffiti, that "a picture is worth a thousand words" (usually subtly misquoted). I know a word that both is worth and describes the thousand features, but it is not one which I can conscientiously use in a family publication.

There is a pattern here which we ignore at our peril. Like bovine spongiform encephalopathy (BSE) which crossed the species barrier from sheep to cattle to wreak havoc on the British beef industry, we have clear evidence of a disease of proliferating features which has already infested both languages and software. Is the Bizarre Species of English which so concerns me evidence of its further spreading to the English language itself ? I don't know, but I fear the worst. Be vigilant !

AND STILL IT CONTINUES : FUZZY SYSTEMS ARRIVE -

What is fuzzy logic ? To one who doesn't know, it's logic which is a bit vague at the edges - and that's a pretty good description of the material studied in the field. For once, we have a good metaphor. We shall pause for a moment to be thankful.

But what's the antonym of "fuzzy" ? Expressions like "clear", "precise", "well marked" come to mind. Whence came "crisp" ? The primary set of meanings given by my dictionary[13] is "firm but brittle, fragile; fresh-looking, cheerful, brisk"; I don't particularly want logic which is plausibly described by those epithets. Then we do get "curt, sharp, decisive", which might be slightly closer to the intention, but are all capable of different interpretations, then finally "*curled, *twisting, rippling" which I certainly don't want. (The asterisk suggests that the usage is archaic, but the taint is there.)

It is also enlightening to consider the antonyms of "crisp". "Fuzzy" does not spring to mind; words like "soggy" and "mushy" carry more conviction. I have certainly read, in journals which should have known better, articles that could fairly be described as soggy logic, and the opposite is certainly to be preferred, but that isn't how the word is usually used. Still, I suppose there's a PhD in soggy logic somewhere; it will come. (Just don't remember that you read it here first.)

- AND HERE'S JAVA.

I thought I had finished, apart from the two notes which follow in the manner of a coda, but now I've started to learn Java. I am forced to use cute "words" like "applet"; I am forced to use "clone" to mean "copy", which is illiterate; I am forced to deal with exceptions, which I have been causing, raising, and handling for years, with methods bizarrely called "throw" and "catch" - which incidentally reintroduces the geographic metaphor of programmes which I thought had gone with "goto". (If they want a new metaphor, I suggest that a disease metaphor might be more appropriate - they could keep "catch", but change "throw" to "infect". Though they'd probably call it "sneeze".) And that's just a few. "Synchronized", of course, is long lost. Should I be grateful that exceptions - not a pretty word, but at least established - aren't called "heyjiminies", or some such amusing alternative ?

Why ? Why be deliberately confusing and misleading ? (Until I found out more about Java, I assumed that an applet was something to do with a Macintosh computer.) Why misuse words ? (I find it offensive to use "clone" when I mean "copy".) Perhaps someone finds it amusing; perhaps they like the jokes. But when you institutionalise jokes in that way, they stop being funny rather soon.

PEOPLE WHO LIVE IN GLASS HOUSES ...

It is not impossible that some could by now be feeling that my polemic is more than a little overstated, and comes ill from one who clings to unusual spelling in programme and disc, and perpetrates words like computist. Well, I can explain everything.

I spell "programme" like that, for all sorts of programme. Computer programmes used to be spelt that way in England, which is where I come from, and I am old enough to have started with that spelling. Later, when people started buying computers instead of building their own, there was an influx of (mainly) IBM manuals. Not surprisingly, these manuals were written in American, a perfectly respectable language in which the spelling "program" is used. Exposure to this usage persuaded some people who probably couldn't spell too well anyway that there was something undignified about keeping the "me" at the end for computer programmes, though not for others, so a quite unnecessary second spelling was introduced into what had been English, which does not need its spelling artificially complicated. Its defenders then contrived to discern some fundamental difference between computer programmes and other sorts which they quoted in support of their choice. It's a difference I have never been able to understand; presumably the Americans either haven't noticed it or don't think it's important, because they use "program" consistently for all sorts; and if it really is something different, then it would surely be more sensible to use a different word rather than to complicate English spelling even more. Bastions of respectability such as the British Computer Society and the Oxford English Dictionary have fallen to "program"; I, with a few other stalwarts, haven't. Time will soon sort us out, but meanwhile we cling to our principles. And the story for "disc" and "disk" is much the same, except that in this case various conversations with people who should know suggest that the confusion is evident even in some parts of America.

The derivation of "computist" is rather different. I use it because I can't find another word which has the meaning I want, which is "someone who computes things with computers, and knows quite a bit about them". Following the usual linguistic patterns, one would call someone who computes things a computer, and indeed before the machines got into the act computers were people who computed, but that's one word the meaning of which has changed beyond redemption. "Computer scientist" won't do because it sounds pretentious, it isn't necessarily accurate, and in any case I don't believe that there is such a thing[6] . "Computer professional" is also pretentious, and doesn't include a lot of amateur practitioners who are at least as knowledgeable. I've seen "computerist", but you don't speak of a "chemistryist" or "chemicalist"; so, by analogy, I'm left with "computist".

So my excuse for my idiosyncrasies is that I don't use them lightly or without consideration; they are the result of more or less rational processes, which convince me even if they find the rest of the ground stony.

SO DOES IT MATTER ?

Yes, it does. Language is our only means of precise communication, and if we're not careful with it we impair our ability to communicate. If we simply invent words as we go along, careless of the possibility that other words with the same or a related meaning might already have been used, or that our new word might be used in a different sense, the result will be confusion.

I have been a computist for about thirty years. I don't for a moment claim to be any sort of universal expert, but my general knowledge isn't too bad. Despite that, I quite frequently come across passages which I don't understand, though they are about topics which I know well enough to be sure that I almost certainly could understand if I knew the language. This phenomenon seems to be particularly severe in abstracts from conference papers, which are intended to be clear enough to summarise the contents of the papers themselves. A common cause is the use of a newly coined technical term without any explanation - or, alternatively, the use of a technical term which I have met before but with a significantly different meaning.

In such circumstances, it is all the more important that we should have a well defined vocabulary with which we can talk and write clearly and unambiguously about computers, in ways which will be comprehensible to people with little supporting knowledge of the technical terms. It would be not unreasonable to hope that those whose positions give them power to mould the language of computing - particularly the larger manufacturers and suppliers of computers and software - would exercise their power responsibly.

In the biblical story of Babel[24] , we are told 'the Lord said, "Behold, they are one people, and they have all one language ... and nothing they propose to do will now be impossible for them. Come, let us go down, and there confuse their language, that they may not understand one another's speech." ' You might have your own views on the theological significance of that passage, but it is clear that the principle that inability to communicate clearly with language renders some activities impossible has been understood for well over 2000 years. It hasn't changed. Why don't we take notice ?

REFERENCES.

1 : *University of Auckland Computer Centre News* **#89/2** (March 1989).

2 : H.G. Baker : "When bad programs happen to good people", *Sigplan Newsletter* **32#3**, 27-31 (March, 1997)

3 : B. Katzenberg, P. Piela : "Work language analysis and the naming problem", *CommACM* **36#4**, 86-92 (June, 1993).

4 : R. Clement : Electronic mail discussion, 1987 (?), quoted in Reference 5.

5 : G.J. Johnson : "Of metaphor and the difficulty of computer discourse", *CommACM* **37#12**, 97-102 (December, 1994).

6 : G.A. Creak : *Does computer science exist ?*, Unpublished Working Note AC65 (19 July 1991).

7 : S. Fennell : *The Tasman Turtle Project - Software Considerations* (07.473 assignment report, Computer Science Department, Auckland University, 1986).

8 : E. Nather : *The Story of Mel, a Real Programmer*, http://www.datamation.com/PlugIn/humor/jargon/jargon_48.html (May 21, 1983).

9 : A. Chandor, J. Graham, R. Williamson : *A dictionary of computers* (Penguin Books, 1970).

10 : D.W. Barron : *Recursive techniques in programming* (MacDonald/Elsevier, 1968).

11 : *IBM Virtual Machine/System Product : CMS User's Guide* (IBM, September 1980), page 317.

12 : *Hypercard User's Guide* (Apple Computer Inc., 1988), page 13. ("Think of the cards in a stack as being looped on a ring; when you reach the end, you start over again.")

13 : A.L. Hayward, J.J. Sparkes : *Cassell's English Dictionary* (Cassell, 19th edition, 1962).

14 : B.E. Martin, C.H. Pedersen, J. Bedford-Roberts : "An object-based taxonomy for distributed computer systems", *IEEE Computer* **24#8**, 17 (August, 1991).

15 : C. Rich : *The Programmer's apprentice* (Addison-Wesley, 1990).

16 : P. Dukas : *The sorcerer's apprentice (L'apprenti sorcier)* (E.F. Kalmus orchestra scores inc., 1933), after a ballad by Goethe.

17 : H.G. Wells : "The truth about Pyecraft", in *Tales of wonder* (Collins, undated).

18 : E.B. Swanson : "System Heroes", *General Systems* **29**, 91 (1974).

19 : *Disk operating system user's guide* (IBM Corporation, first edition, January 1984).

20 : A. Kelley, I. Pohl : *A book on C : an introduction to programming in C* (Benjamin/Cummings, 1984), page 5.

21 : M.C. Board : *New Zealand Herald* (29 November 1991).

22 : B. Lepape, D. Talbot : "HCI : users as the focal point", *Computer Bulletin Series IV* **3#6**, 2-3 (August, 1991).

23 : C. Robbins, in a book review in *Computing* (12 January, 1989).

24 : *The Holy Bible*, Revised Standard Version, Genesis 11 : 6,7.