The Rime of the Ancyent Programmer

Argument: How one programmer's career, set adrift on the raging seas
of technological change, has been tossed by the winds of fate.

There was an ancyent programmer,
a hacker proud was he,
and though well past his prime, he knew
a thing or two, or three.

His hair was in a pony tail,
his bushy beard gone grey,
his face was lined by years of toil
and this he had to say:

My life has been a long strange trip,
so many things I've seen:
technologies both good and bad
and all spots in between.

I started out, now long ago,
on mainframe IBMs,
and many clever programs wrote;
some of them truly gems.

Those were such very diff'rent times.
And coding then was tough.
No screens or mice, (things now so nice),
I tell you it was rough.

The eighty-column card was king,
and cryptic JCL.
You wrote some code, it ran "batch mode".
I tell you it was hell.

Assembler programs I did write,
and complex FORTRAN-4,
but COBOL I refused to learn,
being told it was a bore.

Then came the days of minis, Oh…
My own machine, what joy!
A PDP from DEC I had;
it was a marv'lous toy.

The toggle switches I did use,
with blinding speed I would
machine code programs enter in,
as all true hackers could.

Boot loader ROMs from diodes made,
and home-built D to As.
Both soft and hard ware I'd work on.
Those were such wond'rous days.

So eight inch floppies, paper tapes,
two-megabyte DEC packs,
magnetic, nine-track, half-inch reels,
were storage for my hacks.

Around that time it was I played
my first computer game.
"Colossal Caves" some did it call,
but "ADVENT" its true name.

I managed then to land a job
with a small research team
in a lab doing aerospace work.
It seemed almost a dream.

My main box for the next few years
was nice, I must confess:
an HP-1000 it was,
with a real-time OS.

But soon the minis were passé
and micros became hip.
Assembly language I did learn,
for many a numbered chip:

Sixty eight hundred, Z-eighty,
the sixty five oh two,
(my favourite) sixty eight oh nine.
I learned them through and through.

Embedded systems I'd create –
'twas not a job for fools –
oscilloscopes and logic probes
my sole debugging tools.

With wire-wrap sockets and "perf" boards,
computers I did build.
Then hard-core, real-time software write
and many an EPROM filled.

But then, my boss – a clever man
who had his PHD –
devised a way to measure a
non-linearity.

At first, of course, the skeptics scoffed,
(not being "in the loop"),
but soon the US Air Force came
and funded our small group.

Originally published in *Communications of the ACM*, Volume 50, Issue 5, May 2007.

They bought me a brand new machine,
one hundred K they spent.
An SGI, state-of-the-art,
was what they, to me, sent.

The year was nineteen eighty eight,
I almost lost my nerve.
For at that time, I had to climb
a long, steep, learning curve.

For I was then expected to
not just a user be,
but also a developer
and sys-admin. All three!

vi and C shell, Bourne shell, cron,
lex, yacc, and gcc,
man, make, and diff, grep, find, and biff,
all these were new to me.

So tar and tail, cpio,
compress and uncompress,
uucp and ftp,
and mail all caused me stress.

And if all that weren't quite enough
to make me fume and fret,
I also had to learn to use
this thing called "Internet".

So TCP and UDP
and more I undertook
to learn (to write client-server apps)
from Stevens' exc'llent book.

Those first few years I worked in C,
but soon I learned the knack
of using scripting languages
and never have looked back.

I began writing small shell scripts
and making good progress,
I added bits of sed and awk.
It soon became a mess.

Then one day while on Usenet, I
saw someone recommend
a language that I'd never used
but soon thought a godsend.

The language – it was Perl of course –
I took to right away.
It was such fun, to get things done,
that work seemed more like play.

I used Perl on my SGI,
and also on my Macs,
for data munging and glue code,
and client-server hacks.

And so it was, for a few years,
that things stayed kind of quiet.
But something new was about to
appear and then run riot.

That new idea came from CERN,
the child of Berners-Lee,
I downloaded Mosaic then
and httpd.

I played with them and thought 'ho hum'.
(At first there was no <FORM>.)
But little did I know this was
the calm before the storm.

And soon the year was ninety five –
times I recall so well –
for two new TLAs appeared:
CGI, AOL.

The former seemed a great idea.
I also remember,
the latter caused a thing we called
"eternal September".

"Information Superhighway",
the phrase was all the rage.
Another, less insipid, was
"The Information Age".

"The Web", "The Web" was everywhere,
the press was tickled pink.
"The Web", "The Web", their only care.
It could drive you to drink!

Now after this publicity,
there appeared a whole slough
of self-proclaimed "web programmers"
who didn't have a clue.

About that time I decided
to try Linux "for real".
I had a heavy-duty task
I thought would be ideal.

We wanted to do CFD –
some rather complex codes –
so we built our first cluster then:
twenty-four Alpha nodes.

And Linux did the job quite well;
no real surprises there.
So I began to plot and plan
to use it everywhere.

The next few years brought "browser wars",
the dot-com's rise and fall.
My use, of course, of Open Source
increased throughout it all.

I played around with Java some,
and even Python too,
but they just never seemed to fit
with what I had to do.

So I continue to use Perl
and C when speed is key.
Though neither is now "cutting-edge"
they both work fine for me.

You may have noticed that I just
switched into present tense.
That's 'cause my tale is almost done.
My wrap-up I'll commence…

I've seen many things come and go:
languages, tools and styles,
technologies and companies.
Some bring tears, others, smiles.

It seems the pace is quickening.
there's always something new.
"Change is the only certainty",
and "tempus fugit" too.

So much to learn. So much to know.
So much to do. Alas!
Just take a deep breath and recall
the phrase: "This too, shall pass."

And now the ancyent programmer
sat quiet, looking gruff.
He'd told his tale the best he could;
he hoped it was enough.

His goal of helping others cope,
he thought he had attained.
But if they hadn't learned, at least,
he hoped he'd entertained.

I know the old programmer well;
so well, because you see...
that bearded, balding, code hacker
is none other than me.

Bio:
Stephen B. Jenkins is the senior programmer/analyst at the Aerodynamics Laboratory of the Institute for Aerospace Research, National Research Council Canada. He has written several other 'Geek Travesties' under the pseudonym Erudil.