見本

Prologue

長く活躍できる技術者になろう

――ベテランに学ぶ組み込み技術のＡＢＣ

セサミアン３人組（二上貴夫，坂本直史，山崎辰雄）

「ものごとの基本的なメカニズムに興味を持ち，理解しようとする」のはエンジニアリングの基本だろう．その基本にもっとも近いところにあるのが組み込みシステム設計であると言えるかもしれない．この章では，組み込みソフトウェア開発者として心がけるべきことを，長らく組み込みシステム設計にかかわってきた３人のエンジニアが，自身の体験を踏まえて解説する．（編集部）

夕暮れになるとあちこちでともる灯の中から，にぎやかな声が聞こえてくる，とある４月初旬のこと．各社内で「シーラカンス」，「仙人」，「恐竜」と密かにあだ名される３人が，新橋の焼鳥屋に集まった．

そもそもこの３人は，組み込みソフトウェア技術者や管理者を育成するNPO法人SESSAME[注1]の会合

注1：SESSAMEの正式名称は，「組込みソフトウェア管理者・技術者育成研究会（Society of Embedded Software Skill Acquisition for Managers and Engineers）」である．

で出会った．この団体は，日本の組み込みソフトウェア技術の現状を憂いて，管理者や技術者の教育と育成をめざしている．勤務先も経歴も三者三様の彼らは，もしSESSAMEという活動がなければ，こうして席を同じくすることはなかっただろう．

シーラカンスと仙人は，仙人が企画し，シーラカンスが講師を務めて大盛況だった講習会の帰りである．恐竜の到着を待って入店後，3人でひとしきり話し込んでいる．

シ（シーラカンス）：「なかなか注文取りに来ないね．新人かな？ この店で注文取りが1人というのは厳しいよ．残りのメンバは花見でもしてかぜでもひいたのかな．まだまだ夜は花冷えで寒いからなあ」

シーラカンスが注文取りに声をかけて，オーダを出す．

● プログラミング

仙（仙人）：「新人て言うたら，わたしが新人のときは，OJT（on the job training）という名目のもとに，ろくにプログラミングを教わりませんでしたね．代わりに先輩のプログラムを読まされたけど，ひどいプログラムやった．IBMの大型計算機を使って，入力コマンドを解析するFORTRANのプログラムやったかな？

当時は今のような講習会もなかったし，結局，プログラミングは会社に入ってから自分で勉強しましたね．まあ，配属先がCAD（computer aided design）部門でプログラムの専門家ではなかったさかい，しかたない面もあるんやけど．ソフトウェア開発はだいたい外部に委託していたので，社内に技術がなかったし…」

column 良いソース・コードを読もう

C言語の勉強を始めたころは，Kernighan & Ritchie（C言語を作った人たち）の本(1)を読んで勉強しました（というか，その本しかなかった）．そして，本に載っているプログラムのスタイルをまねてプログラムを作りました．

なぜまねるか．それはKernighanとRitchieが神様だったからです．一種のミーハーです．しかし，この「良いものをまねる」というのはたいせつです．他人のソース・コードを見て良ければまねる，これは決して恥じることではありません．自分の技術を向上させていくひとつの手法です．

技術力アップのためには，良いソース・コードを読みましょう．音楽でも絵画でも，うんちくを聞くよりも，まず自分で聴いて，見て，感じることがたいせつなのといっしょです．自分のソース・コードだけを見て自己満足にひたっていてはいけません．

では，どのようなソース・コードが「良いソース・コード」なのでしょうか．わたしたちからのお勧めは，

恐：「移植を担当していた当時のMIT（Massachusetts Institute of Technology）のXサーバのコードは美しくて勉強になりました．今のXFree86のコードはお勧めしかねます」

仙：「大学では，当時AT & Tが公開していたCコンパイラのソース・コードを輪講して，次にUNIXカーネルのソース・コードを読んだなあ．これは多分に担当教官の趣味で，わたしには理解できなかったけど…．プログラムとはこんなものかという印象だけが残ってます」

と，なにせ太古の話なので，皆さんのお役に立ちません．どんなソフトウェアのソース・コードを見れば勉強になるのかは，まわりの先輩で優秀そうな方に尋ねてみてください．そこで本人の書いたソース・コードが出てくれば，すばらしいのですが….

恐（恐竜）：「わたしもOJTでしたけれど，良い経験をしたな，と思っていますよ．新卒で入社して，いきなり独自プロセッサの論理設計とシミュレーション，マイクロコーディングを担当できたので．プログラミングは独学です．中学生だった1979年から勉強していましたから，大学では仲間うちのだれよりも知っていました」

シ：「OJTのときは，アセンブリ言語とFORTRANのミックスでしごとをさせられたかな．構造化プログラミングが共立出版の『bit』[注2]に出始めたころで，何を勉強したらいいのかわからない時代だった．他人のソース・コードを読んで覚えたなぁ（下掲のコラム「良いソース・コードを読もう」，「書いたプログラムは自分の分身」を参照）」

● 新人教育あれこれ

仙：「昔話はともかくとして，それでは，今の企業の新人教育ではどんなことをやっているんでしょう？」

シ：「うちの会社では，最近こんなことをやりだしたよ．新人と課長連中がチームを組んでの集中講習．要求分析から設計の基本まで教えたうえで，後は自習しながらJava試験[注3]を受けるところまでやる．最後はLEGO MINDSTORMS[注4]のロボットでJavaのコードを動かしてみる（**写真1**）」

仙：「うちは半導体の教育ばかりで，ソフトウェアの教育はないですね．半導体全般，メモリやロジックIC，マイクロプロセッサの基礎，工場実習，販売実習はやるけど．ソフトウェア開発は関連会社が担当しているしね」

column 書いたプログラムは自分の分身

今まで，いろいろな人の書いたプログラムを見てきましたが，プログラムはそれを書いた人の分身のように思います．ちょうど，文章を読むと，それを書いた人の人がらがなんとなくわかるようなものです．

趣味でプログラムを書いているときは自分しか見ないので問題ないのですが，会社の中でプログラムを作ると，そのソース・コードはチーム内（あるいはさらに大きな部門レベル）で共有されます．例えば，自分だけではデバッグできない場合，先輩に指導してもらわなければならないかもしれません．

だから，自分の中途半端な思考を見透かされないという意味でも，他人に見せても恥ずかしくないように，いつも最善のプログラム（そのときの自分が作れるいちばん良いプログラム）を書くように努力しましょう．これを繰り返すことが自分の能力を磨くことになりますし，周りの人も評価してくれることでしょう．

また，ソース・コードは見た目もけっこう重要です．ひと目で「近寄りたくない」と思われるような書きかたをしないように．内容を人に説明しやすいようなプログラムを書くように努力する必要があります．職場で，コーディング・スタイルなどの取り決めがあるかどうかも確認しておかなければなりません．

注2：『bit』とは，コンピュータ・サイエンスの月刊誌である．2001年4月号を最後に休刊となっている．

注3：Java試験とは，米国Sun Microsystems社が提供している，Java言語に関する認定試験のこと．詳しくは，以下のURLを参照．
http://suned.sun.co.jp/JPN/certification/javasc.html

注4：LEGOとは，デンマークLEGO社が発売している，小さなプラスチックのブロックを組み合わせて色々な造形を作ることができるおもちゃ（いわゆるレゴ・ブロック）のこと．LEGO MINDSTORMSとは，レゴ・ブロックにマイコンと組み込みソフトウェアを付加し，自分で組んだプログラムを転送できるようにしたキット商品である．

写真1
LEGO MINDSTORMSを使った講習のようす
（写真提供：WITH http://with.esm.co.jp/）

シ：「全盛期はそれでよかった」

恐：「今いる会社は外資の組み込み系企業ですが，基本的に中途採用なので，教育は個人任せ．放し飼いです．また，以前いた外資系ERP（enterprise resource planning）会社の新人教育はすごかったですよ．1ヵ月100万円の缶詰め講習を受けさせて，最終日に試験をする．受かれば年俸制になるけど，受からなかったら待遇ぼろぼろ」

シ：「じゃ仙人，今日の受講者を試験してみるか．受からなければ代理店からはずす」

● 失敗多き新人時代

仙：「桜が咲いたと言うても，まだまだ冷えるからお鍋を頼みましょ．つくね鍋2人前追加」

シ：「それから日本酒も．一の蔵と…，何にしようか」

仙：「この日本酒はおいしいですよ．まじめなお酒造りをしているんやろう．材料のお米と水，酒造りの技術，そして良いものを作ろうという情熱，これらがそろわないと．ソフトウェア開発といっしょやね．情報工学の基礎的な素養，ソフトウェア作りの技術，そして良いものを作ろうという情熱」

恐：「組み込みソフトウェアの場合，それ以外にもいろいろな知識が必要ですが…．ところで，新人のころ，どんなしごとを任されていましたか？」

仙：「先輩がFORTRANで書いたCADのプログラムの拡張．初めてのまとまったしごとは，プロッタのユーティリティ・ソフトウェアを作るためにデータ・フォーマットを解析したことかな．その後，既存のプログラムを速くするために，開発元の部署に行って，研修も兼ねて2ヵ月間勉強した．研修の後，上司に高速化できそうかと聞かれたから，『作り直したほうがいいですよ』と無謀なことを言ってしもた．既存のコードを見たとき，これをいちいち直していっても，とてもとても…と思っていたので．結局最初から作り直して，20倍から50倍くらい速くなりましたね」

恐：「そういうこと，ありますよね」

仙：「階層設計したLSI回路の階層展開を，前のプログラムはボトムアップに展開していたんですね．回路規模が爆発的に増えてきているところに，フリップフロップを1万個使っていたら，いきなりこれをAND/ORゲートに展開して，その回路をさらに上の階層に伝搬させていく…というようなことをしていた

んですから．これをトップダウン方向の展開にしたら，速くなりました」

シ：「『構造化設計にすると20倍速くなる』というわけだね」

仙：「当時はしんどかったけど，今から考えるとありがたかったのは，上司が真っ赤になるほど報告書を添削してくれたことやね．当時は，上司のきげんの良さそうなときを見図らって，報告書を提出したりしてましたが（p.12のコラム「会社生活を乗りきるための基本の文章術」を参照）」

● 組み込み技術者の誇りを持とう

シ：「この間，SunになぜJavaが良いのか？ と聞いたら，Cは難しいから，だって．Cはポインタを教えても正しく理解してくれる人は少ないし，かといってポインタなしでCで書いても良くない．フレキシブルなJavaのほうが簡単で結果も出る，と言っていたよ」

仙：「すべて配列イメージで扱うから楽やて？ リソースを気にしなくてもいいなら，そう言えるのかもしれないけれど」

シ：「組み込みシステムではそれは適用できない．自分でメモリ・イメージを浮かべられなくちゃ．組み込みソフトウェアは日本で作られるさまざまな機器の競争力の源泉と言ってもいい技術．組み込み技術者は誇りを持って難しいことに挑戦するべきなんじゃないかな（p.13のコラム「組み込み技術者の誇り」を参照）」

シ：「それから，最近聞いた話だけど，うまくいかないのをコンパイラやハードウェアのせいにして『ぼくのせいじゃない』と言うエンジニアがいるんだって．下位層を理解したうえで上位層を作れなくては」

仙：「それはたいせつなことだけれど，実際には難しい．下位層を理解してくれと言っても，なかなかわかってくれへんね．組み込みソフトウェア技術者をめざす新人には，ぜひ，ハードウェアもソフトウェアもわかる技術者をめざしてほしいですね（p.14のコラム「ハードとソフトを両方知ろう」を参照）」

シ：「ほかにも，新しいことにどんどんチャレンジしていってほしいね．10年くらい前に，わたしはある

column 会社生活を乗りきるための基本の文章術

入社したころは，資料を提出するたびに，真っ赤になって上司から返ってきました．それを書き直して提出すると，また真っ赤になって返ってきました．検印をもらうのに，初回の提出から2週間くらいかかるのは普通で，長いものは1ヵ月以上かかりました．そのころはたいへんでしたが，今から考えると，よくめんどうを見てくれたものだと感謝しています．

入社すると，各種報告書に始まり，特許や大小さまざまな申請書，あるいは現在ならメールに至るまで，文章を書く機会が非常に多くなります．名文である必要はありませんが，伝えたいこと，主張したいことを的確に文章にすることが，しごとを手早く片づけることにつながります．

最近，自分の「やったこと」を長々と書いている報告書をよく見かけます．本人は一生懸命書いているのでしょうが，これは，いわば実験データを羅列しているだけのようなものです．やったことから何がわかったのか，あるいはどのようなことが予想されるのかなど，やったしごとに対する自分の考えかたが書かれていない点に問題があります．やったしごとの結果を整理・分析して，まずは人に見てもらう．そして，やったしごとで発見したことや考えたことを人に伝える．このようなあたりまえのことをできない人が多くなっているのでしょうか．

日本語に対する関心が高まっているのか，最近，文章の書きかたに関する著作が多いように思います．何冊か目を通してみてはいかがでしょうか．参考までに，いくつかの書籍を以下に挙げます．

- 清水幾太郎；論文の書き方，岩波書店，1959年3月．
- 木下是雄；理科系の作文技術，中央公論新社，1981年1月．
- 大野 晋；日本語練習帳，岩波書店，1999年1月．

大手のOA機器メーカに通っていたけれど，彼らは新しいソフトウェアの考えかたを取り入れて，高いツールをばんばん買って評価していた．そして，その中から良いものだけ，役に立つものだけを取り入れていた．未来永劫いつまでも良いというものはないのだから，新しい考えかたやアプローチのしかたを取り込もうとする姿勢はたいせつ．若い人たちには，どんどんそういったことにも挑戦してほしい」

● 英語から逃げるな

シ：「この納豆とネギと生卵の入っている小皿の食べかたわかる？ 関西人は納豆食べないか」

仙，恐：「食べますよ．こののりもいっしょに来ていたから挟むのでしょう」

シ：「おっと，やっとつくね鍋が出てきた．この土鍋はうちの土鍋と同じだ」

仙：「土鍋はいいですよね．この土の暖かさが，鍋料理の暖かさとあいまって．鍋料理には，とり鍋，しゃぶしゃぶ，カニ鍋，鴨鍋といろいろ種類があるけど，…」

恐：「技術者に必要なものにもいろいろ種類がありますよね．例えば，技術以外で言うと…」

仙：「まず英語．コンピュータにかかわるなら必須でしょ」

恐：「わたしの場合，新卒で入社した日米合弁のミニコン・メーカでは，技術文書や本社との連絡が全部英語でした．その後国内メーカに移って開発していたときも，ソース・コードのコメントはすべて英語で入れていました．これは中国やフィリピンの技術者をトレーニングするときに役立ちました．仕様書は，英語じゃだれも読めないというので日本語で書きましたけど．

column 組み込み技術者の誇り

パソコンのソフトウェアに誤りがあった場合，「Web上に置いた修正プログラムをダウンロードして直してください」とか，「次のバージョンを買ってください」などと言って，ソフトウェア・ベンダは逃げられます（ソフトウェアについてくる契約書を見ているとおもしろい）．

組み込みソフトウェアの場合，誤りがあったからといって，おいそれとは修正できません．ユーザは，ソフトウェアとしてではなく「もの」として見ています．このため，なんらかの不ぐあいがあると回収騒ぎになります．何回かの携帯電話の回収騒動は，記憶に新しいところです．

出荷台数が多い機器では，回収費用はあっという間に1億円を超えてしまいます．機器によっては，人命にかかわる事故が発生することさえあります．

組み込みソフトウェア技術者ならだれでも，夜中にふっと「あのプログラムのあそこのコードがまちがっているんじゃないか．もう製品に載って出荷されているのに，どうしよう」と冷や汗をかいたことが一度や二度はあると思います．これを気の重い，いやなことと考えず，それだけ信頼性の高いソフトウェアを作っているのだ，と自負できるようになってほしいものです．

仙：「ワークステーションやパソコン，Macintoshといろいろな環境でコーディングするから，日本語のコメントは厳禁．文字コードが複数あるとトラブルの原因になる」

恐：「ニュアンスを表すのに困って，ローマ字でコメントを入れることはあります」

仙：「新技術は英語で発表されるし，Webを見るのにも英語が必要やしね」

シ：「英語ができない，と言うのなら，ソフトウェア技術者をやめたほうがいい．簡単な構文でかまわない．コンピュータ業界の文書は特に簡単だから，それくらいはわかるように努力すべきだよ．昔は，ソフトウェア技術者の要件は『一に体力，二に体力，三四がなくて，五に体力』．これが今は，『知力プラス三つのタイリョク』なんじゃないかな．三つのタイリョクとは，体力，耐力，対異力．最後の『異』は，異人，つまり外国人のこと．人種を問わずコミュニケーションできる力が必要．

column ハードとソフトを両方知ろう

製品を作るとき，ハードウェアとソフトウェアを分けて考えるのが一般的です．例えばパソコンは，ソフトウェアを入れ替えることで汎用的に使えるハードウェアを持っています．

しかし，組み込み製品では，ハードウェアに対してソフトウェアを入れ替えていろいろな用途に使うという考えかたをしません．製品の目的に応じて専用のハードウェアを作り，これまた専用のソフトウェアを作ります．大きな会社でこれらの組み込み製品を開発するときは，ハードウェア設計担当とソフトウェア設計担当に分けて組織が作られています．これに対して，ハードウェアも，ソフトウェアも1人で担当して作ってしまうような組み込み製品もあります．これは比較的小規模な機器で，何かを極限まで追求したいといった用途でよく使われます．その何かというのは，コストを切り詰めて開発したいとか，ハードウェアやソフトウェアの性能を極限まで引き出したいといったことです．

対象システムを1人で設計していると，この機能はハードウェアで実現したほうがいい，もしくはソフトウェアのほうがいいという線引きを1人で考えることができます．ところが，対象システムの開発がハードウェア担当，ソフトウェア担当と分かれてしまうと，この線引きがとたんに難しくなります．なぜなら，ハードウェアはよく知っているけれどソフトウェアはよくわからない，また，逆にソフトウェアはよくわかるけれどハードウェアはさっぱりという技術者が大半を占めるからです．うまく調整すれば最小の労力で最大の効果を発揮できるのに，お互いに何をしているのかわからないために，その実現はきわめて難しくなっているのが現状です．

では，具体的にはどうすればよいのでしょうか．まずは，自分が使うことになっているハードウェアを徹底的に勉強することから始めてみてはいかがでしょうか．「使えればいいや」ではなく，その動作原理や内部構成を理解しましょう．それから，組み込みソフトウェアとは切っても切れない関係にあるマイコンの中身は徹底的に勉強しましょう．周辺回路だけでなくCPUの中身まで．マイコンを骨までしゃぶる，そんなつもりで．

良い技術者，優秀な技術者，長く活躍できる技術者を目ざすなら，ハードウェア開発もソフトウェア開発もわかる技術者になりましょう．

欲を言うと，へたでもいいから日本のことを話せる用意がほしいな．外国人は日本を知らないのがあたりまえだけど，好奇心はある．それをよりどころにして情報交換の方法を会得するのは対異力アップの近道だ．わたしは名ばかりだが茶道が好きで，新人のころは機会があれば茶席へ外人を連れて行った．それから，茶道の英語パンフレットや英語の紹介ビデオなどもけっこうタダで手に入る．これをとにかく読んだな．駅前留学どころか，無料お茶室留学だ（右掲のコラム「情報処理の英語は簡単」を参照）」

● 就職しても勉強しよう

仙：「最近SESSAMEのメーリング・リストでも，工業数学を復習しようという話題が挙がってたね」

恐：「わたしは，数学はぼろぼろでしたが文系の成績が良くて，何とか好きな理系の学部に進めました」

シ：「測定器関連のしごとでは，FFT（fast Fourier transform；高速フーリエ変換）がらみでバタフライ変換をさんざん使ったね」

仙：「それ，わたしのおはこ」

恐：「何のことだか，わたしにはさっぱり…．それでも論理設計，ソフトウェア開発は，かなり深いところを実務で経験してきましたけどね．今は，無線回路関係をマスタしようとしているけれど，数学で立ち往生しています（p.16のコラム「長く使えるものと，すぐ廃れるもの」を参照）」

column 情報処理の英語は簡単

大学のときに，「コンピュータのマニュアルを英語で読めない人はコンピュータを使わないでください」と，最初に先生に言われました．当時は日本語マニュアルなんて少なかったから，ということもありますが，最近でも，やはりそうだなあとあらためて感じています．

コンピュータ関連の書籍やマニュアルで使われる文章には，仮定法や難しい時制が出てくるわけではなく，基本的に簡単な文章です．専門用語が多いのはしかたありませんが，慣れるとへたな和訳より英文のほうが意味が明確です．これはなにも，筆者が英語が得意だからではありません．どちらかというと，嫌いです．しかし，しごとの環境がつねに日本語環境とは限りませんし，最新のソフトウェアを使おうとすると日本語のマニュアルがないことも多いのです．訳本があっても，ひどい日本語なので意味がわからないということもままあり，必要に迫られた結果と言えます．また，Webで検索したときも，ソフトウェア関係の情報は英語のほうが多いのです．

一方，いろいろな作業環境でプログラムを開発していると，文字コードの関係でコメントが読めなかったり，コンパイラによってはすべての種類の日本語コードをサポートしていなくてエラーになったりと，日本語のコメント文のせいで苦労することがあります．

英語に苦手意識を持っている人は，これを機会に，コンピュータ関連の英語を使えるように自己改革を試みたらどうでしょうか．

仙：「最近は，家庭用コンセントに交流100Vが来てることを知らない技術者とか，開発中の製品がおかしなふるまいをすると，調べもせずにリセットする技術者がいるらしい．前者は，技術者以前の問題やね．後者はパソコンの悪い文化が影響しているんや」

恐：「後者に関しては，原因をしつこく追及する意欲がほしいですね」

シ：「トラブル対策だけでなく，与えられたあるいは直面した課題の分析を的確に行い，それに基づいて技術を適用していく姿勢を心がけてほしいね（p.17のコラム「好奇心と基本的なメカニズムの理解がエンジニアリングの基本」を参照）」

column 長く使えるものと，すぐ廃れるもの

10年以上前にマスタした，UNIX，C，Xウィンドウ・システムの知識は今も使えますが，MS-DOS，Windows 3.1，Windows 95などに代表されるパソコン専用のAPI（application programming interface）は，まったく使わなくなりました．新しいものを発表することで以前のものを強制的に陳腐化させるような商品に関連する技術は，マスタしてもむだになってしまう可能性があります．長年にわたって身につけておくもの，業務上短期的でもマスタしなければならないものを見極めて学習しましょう．

また，「大学で勉強してきたことなんて役に立たない」というのは正しい一面もありますが，その教えかたが現実に即していないのでどういうありがたみがあるのかを理解できず，頭を素通りしてしまっているという一面もあるかと思います．どのような組み込みソフトウェアを開発するかによって状況は異なりますが，工業数学が実際のしごとで役立つことはけっこうあります．侮るべからず，工業数学．

仙：「大学のときに，わかりきった実験をすることもたいせつと教わった．寺田寅彦先生が『大通りから路地を見ると行き止まりに見えるが，路地に入ってみなければさらに横道があることに気がつかない．だから，結果のわかっていることでも確かめるということはたいせつなことだ』という主旨の話を書いていると聞いて，感銘を受けましたね．あたりまえのことを確認することが必要…などと言っているわれわれは，生きた化石ですかね（p.18のコラム「〔仙人のつぶやき〕寺田寅彦」を参照）」

仙：「それからさっきも言ったけれど，文書を書くこともたいせつやね．会社にいて何をしているかと言うと，ほとんど文書を書いている．書くもののうちにはメールも入るけれど，メールを書くべきか電話ですますか悩むことがありますね」

恐：「メールだと後に残るからいやだな，と思うときがありますよね」

仙：「内容や書きかたも，プライベートで携帯電話から軽い気持ちで書くメールと違って，会社ではある程度の分別が必要やね」

シ：「でも，組み込みソフトウェア開発は，電子メールとWebを使うようになってから生産性が上がっているよ．いろいろな情報が簡単に入手できる．計測システム開発でも，従来の技術の延長で作ったら4,000万円くらいかかるものが，1/4くらいの費用で作れてしまう．わたしの場合，例えばセンサが欲しいときにWebで情報を集めて良さそうなものを買い，メーカが保証しない範囲でも動くかどうか自分で試してみて，適材適所で使ったりしている．100万台作るもので0.1％の確率で不良が出たらたいへんだよね．でも，50台作るくらいならば，うまく使えばいい．現物で確認して動けばいいんだから」

恐：「使い分けですね」

シ：「そう言えば学校の先生で，上流はやるが，実装はわれわれのしごとではない，と言っていた人がいた．学校の先生は気楽でいいなぁ」

恐：「その話を聞いて思い出しました．ある男がベッドに寝ていて枕元には水差しがある．ふと気づくと，部屋が燃えている．男が数学者だったら火を消すのに必要な水の量を計算してそのまま寝てしまう．物理学者だったら計算して，きっちりの量の水をかけて火を消す．技術屋だったら計算しないで水をかけて火

column 好奇心と基本的なメカニズムの理解がエンジニアリングの基本

さまざまなものごとの基本的なメカニズムに興味を持ち，理解しようとすることはエンジニアリングの基本であり，たいへん重要なことです．ところが最近では，そんなあたりまえのことがあたりまえでないような感じを受けています．家庭用コンセントの交流100Vの件も，笑えない話の例です．

例えば，C言語ひとつ取ってみても，プログラミングを通してコンピュータ（やマイコン）のことがいろいろと透けて見えるはずなのに，文法を知り，プログラムが書けるだけで満足していないでしょうか？

組み込みソフトウェアの場合，この透けて見えるところの知識が重要になってきます．C言語の関数は知っているが，その関数はどのように実装されているのだろうか？ どうして再帰呼び出しのプログラムは動くのだろうか？ どうしてmain関数からプログラムは始まるのか？ パソコン上では，こんなことは知らなくてもプログラムを開発できます．しかし，この「どうして」を追求しないのは，技術者としては問題です．

このごろ，「CPUはどうして動くのか」，「パソコンはどうして動くのか」などというタイトルの本が書店に並んでいますが，このような本を読むまでもなく，自分で調べたり考えたりして自然に知っているというのが理想です．そういった情報は，周りにいっぱい落ちています．技術者なら「どうしてこうなるの？」と感じたら，そのしくみを理解し，「どうすればこれを作れるのだろうか」くらいは考えてほしいものです．

「はい，わかりました」と返事だけ良くても，いっしょにしごとをしていたら，すぐに化けの皮がはがれますよ．

が消えるのを確認して寝る」

（一同笑い）

シ：「んじゃ，仙人は数学者で，わたしは物理学者，恐竜は技術屋だ」

● 選択は自分の意志で

仙：「数学は解が存在することを示すのが重要だけれど，エンジニアリングはだいたいの解，使用できる解

〔仙人のつぶやき〕寺田寅彦

寺田寅彦と聞いて，「天災は忘れたころにやって来る」という格言の作者やとか，夏目漱石の弟子やった科学者やとか，夏目漱石の『吾輩は猫である』の水島寒月のモデルやった人やとわかる人はかなりの御仁だが，普通は知らんわな．少し紹介しよう．

寺田寅彦は1878年生まれの地球物理学者．「X線による結晶解析の研究」は世界に誇るべき研究結果や．また漱石の門下生でもあり，吉村冬彦の筆名で数多くの随筆を残している．その中でわたしが好きな「科学者とあたま」の気に入っているところを引用させてもらおう．短いものやし，ぜひ読んでいただきたい（寺田寅彦の作品は著作権が切れているので，Webの電子図書館「青空文庫」でその一部を読める．「科学者とあたま」も入っておる．長生きはするもんやのう）．

「…いわゆる頭のいい人は，言わば足の早い旅人のようなものである．人より先に人のまだ行かない所へ行き着くこともできる代わりに，途中の道ばたあるいはちょっとしたわき道にある肝心なものを見落とす恐れがある．頭の悪い人，足ののろい人がずっとあとからおくれて来てわけもなくそのだいじな宝物を拾って行く場合がある．

頭のいい人は，言わば富士のすそ野まで来て，そこから頂上をながめただけで，それで富士の全体をのみ込んで東京へ引き返すという心配がある．富士はやはり登ってみなければわからない」

「…頭のいい人には恋ができない．恋は盲目である．科学者になるには自然を恋人としなければならない．自然はやはりその恋人にのみ真心を打ち明けるものである」

が見つかれば，ほかにいくつ解があってもいい．もちろん実際は，コストや信頼性，納期などの兼ね合いから，技術を使い分けるけれど」

恐：「目的のためには手段を選ばずといったところもありますね．必要に迫られて，ベンチマーク対策専用の命令まで作ったこともあります」

仙：「わたしもこの前，平方根計算ルーチンを作りましたね．FFTのプログラムを書いたり，数値計算したり，画像処理したり．顧客からやれと言われたら何でもやる」

シ：「それこそが『組み込み』だね．1年から半年でやっている業務がガラリと変わることもある」

仙：「逆に，ある種類の開発をずっと担当させられている技術者もいますね．10年もやるとつぶしがきかなくなる場合がある．それ以外にしごとがないとか，そのしごとに発展性がなくて新人を入れてもらえないからとか，理由はいろいろやけど」

シ：「ハイ・コスト，ロー・ファンドだね．あいつ以外ではコストがかかるとか言われて．ではもっと良い開発環境をと考えようにも，そもそも資金がない．これを変えられるのは若者パワーだろう」

仙：「いっそ，その製品がなくなってしまえばいいのだけれど，なくならない．会社でメイン・ストリームのしごとにつけなかったときの対応は本人の選択ですね．若者パワーで変えようとするか，会社を見限るか」

恐：「技術の先を見て会社を渡り歩いているわたしも変わり者ですけどね」

シ：「会社との関係は，結婚ではないのだから．自分の意志でどんどん道を切り開いていってほしいな（右掲のコラム「自分のプロジェクトを立ち上げる」を参照）」

column 自分のプロジェクトを立ち上げる

技術者であれば，上から降ってくるしごとだけではなく「わたしはこういう製品を作りたいんだ」という欲求をだれでも持っていると思います．こんなときに，あなたはどうしますか？

正攻法では，商品を企画する部門に異動し，ある程度の経験を積み，マーケティング調査を行って，商品を立案して…と，途方もない時間を費やして，やっと商品になる一歩を踏み出します．こんなことをしていては，作りたかった製品が先にどこかほかの元気な企業から出てしまうかもしれません．

そんなとき，技術者ならではの解決方法があります．「密造酒造り」，あるいは「スカンク・ワーク」と呼ばれる活動です．つまり，会社にないしょでこっそり作ってしまうのです．完成した暁に公開して，周囲を驚かせて製品にします．動くものを突き付けることで社内の反対を抑え，市場に出すところまで最短距離を走るのです．

有名なところでは，NHKの番組「プロジェクトX」で反響を呼んだ，日本ビクターのVHSビデオ・テープ・レコーダ開発があります．かかわる人数が少なければ少ないほど成功する確率が高まります．そんなとき，ソフトウェアだけとか，ハードウェアだけしか知らないというのでは自分のプロジェクトをまとめることができません．また，人を説得し組織する力，プレゼンテーションを行う力など，総合力が求められます．専門ばかと言われることのないよう，力を蓄えることに努力してください．

「密造酒造り」の参考文献としては，『超マシン誕生 ――コンピュータ野郎たちの540日』(2)をお勧めします．今は亡きData General社という新興ミニ・コンピュータ会社が，DEC（Digital Eqipment Corp.）という老舗ミニ・コンピュータ会社に対抗した製品を，新入社員の能力を最大限に引き出して開発したお話です注A．この本はピュリッツァ賞を受賞したことで有名になりました．

注A：ちなみに，恐竜はこのチーム直系の生き残りである．

―＊―

組み込みソフトウェアは日本の産業を支えている主要な技術の一つです．その世界に入った皆さんは希望の星です．「自分に関係のあるのはソフトウェアだけ」，「プログラミングだけ」などと自分の能力を束縛することなく大きく育ってほしい，とわたしたちは考えています．

わたしたちは「生きた化石」ですが，またどこかでお会いする機会もあるかと思います．そのときを楽しみにしています．

参考・引用*文献

(1) Brian W. Kernighan，Dennis M. Ritchie 著，石田晴久 訳；プログラミング言語 C ―― ANSI規格準拠（第2版），共立出版，1989年6月．
(2) Tracy Kidder 著，風間 禎三郎 訳；超マシン誕生 ―― コンピュータ野郎たちの540日，ダイヤモンド社，1982年10月．

セサミアン3人組
組込みソフトウェア管理者・技術者育成研究会（SESSAME）

column ソフトウェアの世界は，まだまだLSI発明以前

ハードウェアを担当してきた方々からすると信じられないことかもしれませんが，ソフトウェア開発の世界は，コンピュータが発明されてから50年以上にもなるのに，ハードウェアに例えるとLSIが発明される以前の状態です．せいぜいTTL（transistor transistor logic）ICが作られた程度でしょうか．

CPUは，メモリから読み込んだ情報を命令として解釈して実行します．メモリには，命令があってもいいし，データ（文字や画像，音などを表すもの）があってもかまいません．CPUがメモリから取り出して実行する命令（言語）を「機械語」といいます．機械語は，CPUを作るメーカごとに異なっています[注B]．この「機械語」は，ハードウェアに例えると，抵抗，コンデンサ，コイルなどの基本部品に相当します．

「機械語」は，数値の情報（かっこよく言うと「特定のビット・パターン」．01010011 00110110…など）にすぎないので，そのままでは扱いにくくてしかたがありません．そこで機械語に1対1で対応する言語として「アセンブリ言語」が作られました．機械語レベルのプログラミングはアセンブリ言語で記述して，アセンブラというプログラムで機械語に変換します．今どき，抵抗やコンデンサ，コイルだけでハードウェアを作ることがないように，機械語だけでソフトウェアを作ることはきわめて少なく，あっても限られた人だけが扱うようになりました．

現在，アセンブリ言語に代わって組み込みソフトウェア開発に広く使われている言語がC言語です．C言語で記述して，Cコンパイラというプログラムで機械語に変換します．C言語を使うことで，完全ではありませんがCPUに依存しないプログラムを作ることが可能となりました．

今や携帯電話などに組み込まれるソフトウェアは，数百万行のプログラム・コードで作られています．数百万個のTTL ICが使われていて，大量のジャンパ線が飛び交っているハードウェアを想像してみてください．こわいですね．一発で動いたら奇跡でしょう．

2001年には携帯電話メーカ各社が，ソフトウェアの誤りを原因とする製品回収を相次いで発表し，新聞紙上をにぎわせました．ハードウェア開発が，基本部品の組み合わせからLSIを使った設計に切り替わったように，組み込みソフトウェア開発でもようやく，携帯電話のブラウザなどにパッケージ・モジュールを使うようになりました．買ってきたLSIをつないで使うほど簡単ではありませんが，ソフトウェア開発も，しだいにハードウェア開発で使われている考えかたに近づきつつあります．

注B：CPUには，最近，携帯電話に使われることが多いARMや，家庭用ゲーム機に使われているMIPS，ルネサス テクノロジのSHとM16C，米国Intel社のPentiumなどがある．これらがそれぞれ独自の機械語を持っているので，CPUを別のものに変更するというのは，とても大きな決断になる．ハードウェア開発では，CPUのハードウェア仕様書を読んで端子をつなげばひとまず終わりかもしれないが，ソフトウェアは，その開発環境も含めて総入れ替えになる．今まで蓄積してきたプログラムもそのままでは使えなくなる．