TSG

Theoretical Science Group

理論科学グループ

部報 200 号
— 夏休み号 —

目　次

ドキュメント夏合宿 (おい)

かも

今年の夏合宿では，長野県の野沢温泉に行ってきました。

合宿の概要

日程　7/31(水) 夕〜8/3(土) (三泊四日)
費用　29,000 円 + 現地徴収 4,000 円
参加者　のべ 18 人
経澤 (8/2 夕から参加) 須磨 多賀 西澤 大岩 岡村 鴨島 木原
高野 (8/2 朝まで) 富樫 中村 (誠) 松村 馬本 黒川 丹下
寺内 保原 宮崎

経過

7/31(水)

出発当日である。

昼過ぎからみんなわらわらと 305 に集まり始める。出発まで時間があったので、僕の持っていったサターンをやったりして時間をつぶす。6:15 にバスで駒場の裏門を出発して、途中休憩を挟んで[1] 23:00 ころ宿に到着した。野沢は東京とは比べ物にならないほどの涼しさである。部屋は 4 部屋使えたが、須磨さん持参の「僕の地球を守って」を読みふける部屋、サターンの部屋、「須磨の 300 番問題」の部屋、疲れて寝ちゃった部屋に分かれたようであった。僕は宿のぬるい風呂に入った後、サターン部屋で木原君、富樫君、松村君や馬本君たちと「デカスリート[2]」にはまってしまい、結局寝たのは 2 時ころであった。なんと 300 番問題の部屋では朝まで寝ないで問題を解いていたそうである。

[1] 須磨さんは途中で休憩をとった談合坂 S.A. から italk していた (^^;
[2] その名の通り十種競技のゲーム

8/1(木)

8:00 に朝食。300 番問題を解いていた人たちは、疲れ果てて午前中ずっと寝ていたらしい。

午後からは、みんなでゴンドラで山の上に登った。山の上はとても涼しく、秋のようであった。ゴンドラで登った地点には、冬のスキー場のゲレンデを利用したプラスノスキー (普通のスキー板と同じ形状の板で滑るグラススキーみたいなもの) や、そりで山の中のコースを滑り降りるスーパースライダーなどの施設があり、特にスーパースライダーの方は受けがよかったようである。馬本君はスーパースライダーでスピードを出しすぎてころんでしまったそうだ。また近くの毛無山 (けなしやま) という山に登った人もいたようであった。

3 時半ころに山から降りて宿に戻った。夕食まで時間があったので、松村君や高野君

たちが帰りのゴンドラの中でなぜか話題に登った昔懐かしミニ四駆を買ってきた (^^; 何でも今は子供たちの間で再びミニ四駆がはやっているらしい。というわけで、彼らが組み立てて走らせるのをみていたが、僕が小学生のころにつくったようなマシンとは大違いで、ボディ各部に軽量化が施されていたり、ホイールにはデフギヤがついていたり、シャフトが中空になっていたりで、とにかく速かった。ミニ四駆にここまで技術をつぎこんでしまっていいものなのであろうか (^^;

夕食の後は、花火の買い出しもかねて温泉に入りにいった。野沢温泉郷には、13 箇所の外湯と呼ばれる共同浴場があり、宿泊客は無料で入ることができる。まずは宿から一番近い大湯という温泉に入りにいった。しかし、大湯にはぬる湯とあつ湯という二つの浴槽があったのだが、ぬる湯といってもこれは熱湯コマーシャルかと思うほどの熱さで、ましてやあつ湯に至っては足すらつけることもできなかった。結局須磨さんはそこで撤退し、残った人々も 4 箇所くらいしかまわれなかった。[3)]

10 時ころからは、花火をやった。昼のように明るくなるという宣伝文句の「レーザー」は、本当に明るかった。バケツを準備するのを忘れたので、結局半分くらい消費したところで残りは翌日にとっておくことにした。その後突発的に飲み会になり、やがて怒濤のモノポリー大会になだれ込んでいった。僕は酔っ払ってしまい、早々に寝てしまったのだが、モノポリー大会は朝まで続いたようで、モノポリー大会の部屋で寝ていた僕は何度も目を覚ましてうるさいうるさいと文句を言っていたらしい。

8/2(金)

朝食の後、高野君が東京に用事があるということで帰っていった。[4)] 前の晩にモノポリーをしていた人達は案の定午前中ずっと寝ていた。午後からは、野沢温泉アリーナという国際会議場も備えたスポーツ施設[5)] の屋内プールに泳ぎにいった。流れるプールや波の出るプール、ウオータースライダーなどがあったのだが、屋内プールなのであまり規模は大きくなかった。しかしやはり野沢温泉ということで温泉が施設内につくられていた。

プールから帰ってくると、遅れて参加の経澤さんが到着しておられた。夕食の後は、宿の食堂でコンパが行われた。

その後は、前日の残りの花火を消費し、部屋に戻って前日に続いてモノポリー大会が行われた。経澤さんは、どうやらサターン版の「ときめきメモリアル」を持ってきたようで、朝方までプレイしてエンディング寸前までいったらしい。僕のサターンには、今でもその時のセーブデータが残っている (笑) [6)]

---

3) 残りの温泉もかなり熱かった (^^;

4) 8/3 からのコミケに行くためだったそうだ

5) 野沢温泉スキー場は、長野オリンピックのバイアスロンの会場になっているそうで、その関係で国際会議場があるのだろうか

6) ちなみに主人公には僕の名前をつけたらしい (^^; 全くひどい話だ (^^;

8/3(土)

午前中に、土産を買いに出かけた。モノポリー組は、また午前中ずっと寝ていたらしい。

昼食をとって、1時に宿を出発した。帰りのバスではやはりみんな疲れていたようで、口数が少なかった。途中バスが渋滞に巻き込まれたりして、解散場所の新宿西口についたときには7時になっていた。

夏合宿の舞台裏

ZERO

***この文章はあくまでも ZERO の視点で書かれています (笑)**

3 月某日 コンパ委員の内数名が合宿の準備を何もしていないのに気付く。焦る。焦るが、じたばたしても仕方がないので (ぉ 次のコンパ (新歓コンパ) の時にアンケートを採るという方向性で進めることにする。[1)]

4 月 10 日 サークルオリ初日。見知らぬおぢさんが「あのぉ〜、TSG はこちらでしょおかぁ」といきなりやってくる。他にコンパ委員がおらず、何故か ZERO が応対する羽目になる。この後しばらく、ZERO が窓口役になる。[2)]

おぢさんの話では、おぢさんは磐梯の宿の人で、10 年ほど前にその宿で TSG が合宿を行い、記念ということでロゴ入りのトレーナーだか T シャツを置いてきたとのこと。[3)] パンフレットを見る限り、悪くは無さそうだ。ここにしたれ、と脳裏に陰謀が渦巻く (笑)

取りあえず、後でまた連絡するという話になりその場はそれでお引き取りいただく。

4 月 11 日 今度は旅行代理店の人がやってくる。かもと ZERO が応対する。いろいろと良さそうなことを言っていくが、2 人の心は昨日のおぢさんの宿に奪われ、ほとんど意に介しない。くそー、SKY DUAL 良いペースで来ていたのにー (ぉ

名刺と、どこぞの遊園地の割引券を置いておぢさん去る。かもの様子がおかしい。必死に笑いをこらえている。訳を聞くと「お前ら、気付かんかったのか? あのおっさん○ラだぞ。」 その場がその後どうなったのかは言うまでもないことだろう。

4 月 19 日 今度はわざわざ部室まで白馬のペンションの経営者とおぼしき人がやってくる。いろいろ言って行くが、磐梯に条件がかなわない。しかも、このおぢさんのおかげでバイトに遅刻する羽目になる (=_=)

4 月?日 磐梯のホテルから電話が来る。まだ決めるのには時期尚早なので取りあえず色好い返事だけして「お世話になる場合は 5 月×日までに御電話いたします。」などと言ったがその後電話などはしなかった (悪)

おぢさん、ゴメンね (笑)

[1)] 去年は既に新歓コンパでは参加/不参加のアンケートだったような... (^^;

[2)] ZERO は行けなくなったので途中からかもに代わる。

[3)] どなたか、参加した方... いらっしゃらないか (笑) 写真とかあると愉快なんですけどね。

夏合宿の舞台裏

5 月 16 日 新歓コンパの前々日、フェニックス観光とやらから電話が来る。しかも、わたる、かもからたらいまわしで (笑) 取りあえず電話代は向こう持ちなので、追試も終わったことだし話だけは聞いてみる (笑)

すると、ぎりぎりまで人数変更オーケーだの客へのアンケートを繰り返して斡旋する宿の淘汰を行っているだの、結構いい条件を持ってくる。学生向けなので料金も格安だとか。取りあえず、翌週の月曜日に会う約束をして電話を切る。

5 月 18 日 新歓コンパ。合宿の希望地のアンケートを採る。海山離島でアンケートを採るが... 離島を選択肢に入れたのは不覚であった。結果は海 3 山 6 離島 17 という、最悪の結果になる。ZERO がやや切れ、sigma さんになだめられるという失態を犯した (^^;

5 月 20 日 フェニックス観光へかも、きりもみ、窓明、ZERO の 4 人で乗り込む。相手は新入社員であったのに、トントンとうまく話が進み (うまく乗せられた、というつっこみはなしね (^^;)、アンケートの結果を反映せずにコンパ委員の全員一致で野沢に決定。[4] (*4)

その後、この旅行会社とちょくちょく連絡を取りながら合宿を企画していくことになる。

7 月 13 日 道を歩いていると、かもが向こうから苦笑いを浮かべてやってくる。

「試験 31 日までだってー。教務課さいてー」

合宿の出発予定時刻を夜に変更。夜行バスでの野沢入りというプランになる。

7 月 15 日 旅行会社から「夜行バスだと割高になるので夕方出発ではどうか?」と電話が来る。初めからそう言えー (=_=) 仕方がないので夜中にいぬ。に書込をしてほぼ事後承諾のような形で夕方出発に変更する。結果的にこの判断は当たりだったのだろうか?

このとき一緒に最終的な参加者を発表するが、その後変更が相次ぎ結構焦る。この観光会社チョイスして良かった、と思った数少ない点 (笑)[5]

しかし、バス代が人数割りなので何度も費用を再計算する羽目になる。それでも追っつかなくなり、別途徴収という手段に出る羽目になる (T-T)

7 月 26 日 ばんだい君が消息不明になる。ぴんちぴんち (T-T) 結局「レスがなければキャンセル」という緊急手段に出るが、これといったトラブルもなく胸をなで下ろす。[6]

---

[4] 離島は大島くらいしかなく、去年も大島だったので回避しました。決して独断専行ではありませんので。為念。

[5] かもの談話では宿はなかなか良かったとのこと。冬のスキーも...(ぉ

[6] ばんだい君は自然気胸で入院していたとのこと。長期じゃなくて良かったですね。

7 月 31 日 いよいよ出発日。いろいろあったがここまで来たかと思うと ZERO はホッとする (参加しないから (^^;;)。

しかし、TSG はそんなに甘いところではなかった。集合時間も近いというのにまだ数名がやってこない。かもがどこかへ走っていく。たけしまとげると ZERO も学館付近をうろうろして待つ。[7) 8)]

6 時直前になってようやく 1 名の欠員 (当日キャンセル =_=) を除いて揃う。

取りあえず、出発。たけしまと ZERO が見送る。聞こえないのを良いことに○集長に「Fe (謎)〜」などと言って遊ぶ (編○長、怒っちゃいや〜ん (ぉ)。しかし、道が詰まっていてバスが出ない。見送る方は凄く間抜けだ ;_; 早く出ろーと念じるが、願いもむなしくまだ出ない。しくしく。

バスがようやく出る。これで一仕事終わった。[9) 10)]

情報棟に行き、italk に参加する。寂れている (笑) 7 時頃であろうか、誰か来た。誰だろと思ったら...

[あぶら@談合坂] ... (爆)[11)]

おしまい。

おまけ: 合宿の費用が少々余りました。細かい会計処理がまだなので金額は確定できませんが、一人頭 400 円程度の返金になりそうです。返金方法などは追っていぬ。BBS などでお伝えします。

---

7) 確かに集合を早めに設定しなかった方も悪いですけど、みなさんもうちょっと早く来ません?

8) このときげるに「プリンシェイク」とかいうジュース (?) を差し入れ、3 人で飲むが、「うまい」と言ったのは... げるだけだった (笑)

9) 実はまだこの後当日キャンセルの人からの集金と余剰に集金した人への返金つまり会計業務が残っている ;_;

10) このあとたけしまは庶務の職務を全うすべく、学館大掃除の前日準備に向かった。ご苦労様。

11) しかし、筆者の確認しているだけでも合宿中に油すましさんと高野商店がいぬ。に現れている (笑)

鉄道メモリアル

Tellur

鉄道メモリアル、といっても鉄道雑誌の一種ではありません。巷には鉄道ファンとか鉄道ジャーナルとか、鉄道ピクトリアルとか Rail Magazine などいろいろありますけど。かといって、「たびてつ友の会[1]」でもありません。いくらかっこよくても、あんなこっぱずかしいラブストーリーが旅先で待っているはずがありません。これは合宿に途中参加したが為に、昼は鉄となり、夜はメモラー[2] と化した哀れな部員の物語なのです。

なんか註[3] が巨大になってしまいましたが、まあ気にしないで下さい。

出発したのは 8 月 2 日の朝でした。本当は早朝に出て昼過ぎに着くつもりでしたが、前日にお通夜の後の厄落としと称して、新宿で 11 時頃までどんちゃん騒ぎをしていたもので、早朝にはとても起きられなかったのです。一泊分の荷物を鞄にほうりこんでから、何か精神汚染できそうな物はないかと部屋を見回すと、封印されていた危険物の土星[4] 版が目についたので鞄に忍ばせました。電話で幹事に「夕方 6 時頃そちらに着く」と伝えてから、10 時過ぎに家を出ました。

京王線の下高井戸駅から 10:27 発の各駅停車に乗って新宿に向かいます。とりあえず上野に出ようと思いまして、新宿で山の手線の外回りに乗りました。田端に着いたところ、京浜東北線の快速が来るというアナウンスが聞こえたので、あわてて飛び降りて乗り換えます。日中の快速はやはり足が速く、山の手線より先に上野に着きました。

上野で急いで買ったのが、特急あさまの指定席[5] です。さすがに夏休みともなると自由席は家族連れで混んでいるようなので、直前のキャンセル狙いで窓口の横の指定券自動販売機で調べてみると、喫煙席ながら 11:30 発のあさま 13 号の席を取ることができました。あさまが無理なら上越新幹線で湯沢に出てバスで山越えしようか[6] とも考えていたので、席が取れてからまた慌てて戸狩野沢温泉までの学割乗車券を買いました。際どいことに、改札を通ったのは発車まであと 10 分という時でした。席の番号は 5 号車の 13C、ホームに行ってみると何と車両はモハ 189-1、横軽越えの為に作られた 189 系[7] の最初の編成に組み込まれた由緒ある車両です。アコモ改造[8] を受けてデラックス車両になっていまして、座席は比較的大きく足も伸ばせます。折角だからと、しっかり写真

[1] ヤングアニマルに連載されているそうで。

[2] ときめきメモリアルにはまってしまった人のこと。さらに特定のキャラに壊れて、そのキャラの悪口でも言おうものなら怒りだす人のことはメモリアンというそうな。

[3] 筆者の独断と偏見だらけである。

[4] SegaSaturn のこと。HiSaturn や VSaturn でも可。でもこれを「ほたる」と呼ぶ人は重傷だ。対称型デュアルプロセッサ構成の熱いマシン。

[5] 繁忙期なので自由席の 700 円増し。学割の割引分がふっ飛んだ (;_;)

[6] あずさで中央線経由で行くのは遠回りなので、始めから除外したが、振り子特急には乗りたかった…

[7] 碓氷峠を越えられるほかは、ちょっと前の房総特急に使われていた 183 系とたいした違いはない。

[8] 鉄は「内装をやり変えた」のをこういう風に表現する。

も撮ってしまいました。持ってきたカメラはCONTAX[9] G1。一眼レフではないのですが、比較的大きくて明るいレンズを付けることができるAF機能付きのカメラです。で、今回の旅行ではこのカメラで、あちこちでぱちぱちと写真を撮りまくることになるわけです。

乗ってから時刻表を見て気がついたのですが、この特急は戸狩野沢温泉に出るのに一番乗り継ぎの良い列車でした。大宮を11:52に出発して間もなく、売り子が弁当を売りに来ましたので、折角峠越えをするのだからと釜飯を買ったところ、本当に素焼きの釜が付いてきました。この釜は、宿についてからきれいに洗い、乾かしてから鞄の中に大事にしまわれる事となったのです。こんな物を持ち帰ってどうするというのでしょうか…[10]

大宮と高崎の間では、熊谷と本庄に停まりました。ちょっと意外だったのが本庄で降りる客が多かったことです。確かにここは新幹線が通過しない所ですけど、なぜ昼間の特急であさまになってしまうんでしょうねえ[11]。

横川に13:00に到着、発車するまでの間に上りのあさまや駅の看板などを撮ってしまいました。7分の停車時間の間に、機関車の重連を後ろにつなげます。さらに空気バネ[12]の空気を抜いてしまいました。いよいよ碓氷峠越えです。峠といっても鉄道の場合は、一方的な登り坂が待っているのですが。この峠の区間は北陸新幹線が開通すると廃止されるので、一度乗ってみたかった区間でした。

ここの勾配は地下鉄もびっくりの急坂が続きます。東京でこれに匹敵するのは、目蒲線の目黒駅へのアプローチ[13]ぐらいでしょうか。

その坂を列車は、サスが効かないのでびりびり震えながら登っていきます。途中の信号所では崩れかかった機関庫などが見えました。

軽井沢には13:24に着きました。隣には工事中の北陸新幹線のホームが並んでいます。ここでもちゃっかり特急の最後尾や機関車の写真を撮ってしまいました。それまでは家族連れや初老の一人旅が多かったのですが、軽井沢でどっと降りてしまいましたので、残りはほとんどがビジネスマン風の客となりました。この先は急行のように小諸、上田、戸倉、篠ノ井と停まっていきます。そして14:22に長野に着きました。軽井沢–長野のほとんどの区間で北陸新幹線の高架の工事は終わっています。オリンピックの力は本当に偉大ですね。

長野で7分待ちで飯山線の鈍行に接続ですが、戸狩野沢温泉行き139Dはキハ52[14]の非冷房車2両編成でした[15]。山の中なら冷房無しでもいいのですが、市内を走るのはちょっとつらそうです。車内を見渡すと、若者のグループが多いようです。列車は長野を

9) 今は京セラが持っているブランド。
10) いまだに使われないまま台所の横に置いてある。
11) 上野発11時台には新特急は一本も走っていないのであった。
12) いい電車はエアサスを入れている。古い国電はコイルバネだ。
13) ここももうすぐ地下化されておもてから消えてしまう。ぐっすん。
14) かなり古い普通列車用のディーゼルカー。エンジンは国鉄標準型を二機積んでいるので、パワー不足ということはない。
15) 飯山線は一応キハ56のような冷房付きの急行型車両も走っている。優等列車はないが。

出て豊野[16] まで信越線を走ります。右手の遥か彼方には新幹線の高架が続いています。これは車両基地が長野のだいぶ北に作られているためですが、北陸新幹線の延伸を睨んでの事でしょう。豊野で飯山線に入り、谷間を進んでいきます。そして 15:29 に戸狩野沢温泉に着きました。バスの発車まで 7 分あったので、またも列車や駅舎などをぱちぱち撮ってました。バスにゆられる事 20 分弱、温泉街の中心から結構急な坂道を上って、4 時過ぎにみんなの泊まるやまや山荘に到着しました。

部屋には部員が半分くらいしかいないようでした。聞いてみると、他の人はプール[17] に行ったとのこと。多くの人が寝ていましたが、一部はゲームをやっていました。そういう訳で、食事までの一時をゲームを見ながら過ごしていたのです。

プレステ[18] では鉄拳シリーズ[19] が動いていました。相変わらず平八[20] の空中コンボ[21] が炸裂しています。隣の部屋では、土星で映画版のスト II という恐ろしいゲーム[22] が動いていました。でもすぐにデカスリートに代わりました。デカスリートでは、世界記録を出した瞬間を写真に収めてしまいました。この後さらに、ヴァンパイアハンター[23] に移ったようです。夕食の後はひでき君がプレステ版のグラディウス[24] で名人芸を披露してくれました。

で、宴会まで少々時間があったので、そろそろ部員の精神汚染を謀らねばならないと思い、隙を見てとうとう某デラックス版の封印を解いてしまいました。まずは音楽トラック 3 に入っている謎のおまけ[25] を聴かせました。そして、オープニングのビデオ[26] も見せましたが、好き好んで自ら汚染される剛の者はさすがにいなかったようです。

そして宴会が下の食堂で始まりましたが、私は酒は少しにしてソフトドリンクばかり飲んでいました。そして客室にて二次会が始まり、モノポリーが展開された頃に、ついに恐怖の恋愛シミュレーションゲームを始める事と相成ったわけであります。

折角ですから、土星様の持ち主にちなんで、主人公を次のように設定しました。

名前 鴨島 潤

あだ名 かも

誕生日 6 月 18 日

16) 夜行のスキーバスで来る時は、この近くのドライブインが最後の休憩となる。
17) 浅かったらしい。幼もえな人が危険になるような所だったらしい (謎)
18) PlayStation。ソニーコンピュータエンターテインメントの出したゲーム機。得意技はポリゴンぐりぐりと JPEG 展開らしい。メーカー推奨の略称は PS。
19) ナムコの出した格闘ゲーム。濃い。
20) 1 の親玉。2 の主人公。破壊力抜群のカミナリおやぢ。
21) ふっ飛ばされた相手に畳み掛けるように攻撃する極悪な連携技。これを知らずに鉄拳をやるのはきつい。
22) 映画版の怪しげな人物がくるくる回る迷作。
23) カプコンのだした格闘ゲーム。怪物どもが理不尽な攻撃を披露する。
24) 筆者はちちびんたリカの方が本家だと思っていたくらいグラディウスに疎いので、II なのか III なのか未だに良く判らない。I は X68000 でやって、辛い思いをした。処理落ちしないんだもの。
25) 出てくるのは風間準ではない (激謎)
26) タツノコに委託した物らしく、キャラの感じが少々違うので、悪く言う人もいるようだ。

血液型 B

お気づきの方も多いでしょうが、部員の誰かさんそのものですね。ついでにヒロインである幼馴染みの藤崎詩織嬢[27] の設定もこのようにしました。

誕生日 6 月 18 日 (部活は吹奏楽部[28])

血液型 AB[29]

そして、新学期が始まってすぐにバスケ部[30] に入部した訳です。これも誰かさんの高校時代にあわせていたりします。本人は結構悶えていたようですね。

最終的にどういうエンディングを迎えるのを目標にするかは、早い段階で決めたほうがいいのですが、これは少々迷いました。女々しい野郎どもの詩[31] をカラオケで[32] 聴こうとか、伊集院[33] に電話をかけまくろう[34] とか、ひたすら寒いプレイをして館林[35] 告白を狙うかとか、いろいろ考えましたが、周りから

「まともに詩織を狙え」

と言われてしまいました。

このゲームでは、主人公のパラメータによって色々な娘が出てくる[36] ので、注意深く調整しないと、全員出してひどい目にあう[37] ことになります。大体初心者は 5〜6 人に抑えた方が良いでしょう。

で、今回はどうだったかというと、まずは主人公が運動部[38] だからあっさり虹野沙希嬢[39] が登場しました[40]。次に、一年目のクリスマスで清川望嬢[41] が登場しました。

登場した女の子が少ない時は、新たにクリスマスパーティで登場することがあります。誰が現れるかは確率的な要素が強いのですが、モノポリーをやりながらゲームを見ていた松村氏[42] の話では、主人公のパラメータも影響するとか。そして一年目のバレンタインで美樹原愛嬢[43] が登場しました。詩織狙いで行って、部活などで活躍する限り、登場

[27] ヘアバンド集めと音楽鑑賞を趣味とする。文武両道で才色兼備で理想が高く、少々子供っぽくて性格が悪い。やはり人気高し。
[28] PS や Saturn では (月+日×3) mod 11 で詩織の部活が決まる。
[29] B 型と相性がいいんだそうだ。
[30] 実は、この部でしか見られない絵もあって、それは夏合宿の最後の夜の (以下自粛)
[31] バッドエンディングで流れる曲。なぜか出来が良い。
[32] ある寒い条件を満たすと、カラオケバージョンが流れる。ちゃんと別の AIFF に入っている。
[33] ライバルらしい。名前はレイ。きらめき高校の理事長の孫。よくいやみを言う。
[34] かけすぎるとメッセージも変われば、変なイベントも起きる。75 回先に何にでくわすかは公然の秘密。
[35] 名前は見晴。その名のとおり主人公を監視している、いや単におっかけをしている。サターン版ではかなりイベントが増えた。鉄山靠の使い手の一人 (謎)
[36] 特定の部活をするのが出現条件になっていることもある。
[37] Yu 君は初めてやった時に全員出したのに閣下が告白したそうで、すごいものだ。
[38] 運動をすると、根性も少し上がる。
[39] 根性のある人を好く。弁当が有名。ころぶ人多し。
[40] 詩織以外が一人以下しかいない時の登場条件はゆるい。
[41] 超人。さっぱりしてるが惚れっぽい。花を愛でる。水泳部の特待生。
[42] 鉄拳分科会の親玉の一人。今回筋金入りのメモラーであることが発覚。
[43] めぐみと読ます。詩織の友人。通称ヘルメット頭。爆弾処理を妨害したり、他の娘を押しのけて告白したりするので嫌う人が多いが、筆者はわりといいなと思っていたりして…

は避けられないようです。二年目の頭で早乙女優美[44] が登場するのは確定です。主人公の親友である好雄[45] の妹ですから。登場した女の子はこの 5 人のまま話は進行していきます。

一緒に下校するのは詩織と虹野[46] だけにしました。一緒に下校すると他の娘の傷心度[47] が少しずつ上がっていきますので、あまり八方美人するのも考え物です。

それと、好感度が高くてかつ傷心度が低い時に、女の子からデートに誘ってくることがあります。松村氏から「本命以外からデートに誘われたらリセット作戦」という恐ろしい作戦を聞きました。一度目はいいけど、さらに続けて本命以外から誘われるのなら、すぐリセットしてやり直すというものです。実行してみると優美や清川さんのせいでリセットするわするわ、20 分ぐらいゲームの進行が停まっていたようです。まあ本命以外のときめき度を上げると後が厄介なので、デートでも少々冷たくあしらうようにしました。例えば優美にショッピングに誘われた時はわざとジャンク屋に連れていって[48]、さらに「サイバーな感じがたまらないぜ」なんて答えて泣かせるなどというのは、極端な例ですが、できるだけベストな受け答えをしないように注意しました。ちょっとびっくりしたのが、一度館林が下校の時に話し掛けてきたことです。これはサターン版で新たに入ったイベントで、館林の出現する場所[49] が他のに比べて相当増えているようです。

一年目は適当にやっていたので、勉強系のパラメータの上がりは悪かったようです。そこで二年目は詩織とデートをしつつ[50] 気合を入れてパラメータ調整をし、終りごろには文系のパラメータを相当高くしました。

二年目でもうひとつ、松村氏の「女の子が登場するぎりぎりまで平日コマンドで勉強する作戦[51]」はものすごい物がありました。平日が確かに有効に使えるのですが、誰か新たに登場した所でリセット[52] しなければなりません。そして、理系学習のパラメータをあげている時は、つい紐緒結奈嬢[53] を出してしまい、多彩な登場パターンに見とれては何度もリセットしていたのでした。こういう事をしているとどんどん時間は経っていきます。しかしサターン版[54] では、深夜になると開始時の女の子のメッセージが変わります。「無理をしないで」なんて言ってくれますが、合宿でわざわざやっている人にゆっても無駄ですね。

---

44) 強烈なこどもっぽさで他を圧倒する。バスケ部員。
45) ナイスガイ。愛の伝導師。女の子の情報なら任してくれだそうだ。
46) なぜ虹野のご機嫌をとったのか、それは虹野燃えな松村氏がいたからだけど。
47) だいぶ傷つくといわゆる爆弾点火状態になり、鎮めるためにデートする必要がある。爆発すればみんなから嫌われてしまう。
48) なぜ、向こうから誘っているのにブティック・小物・ジャンク屋の三択がでるのだ？
49) いままでは、学校の廊下、誰かの春か海でのデートに乱入、謎の留守電、中央公園の 222 事件しかイベントがなかったはず。背景にこっそり姿を見せている事もあるが…
50) どうせなら本命とは一ヶ月に一回以上したいものだ。
51) 休日に頑張る限り、虹野・美樹原以外の女の子は新たに登場しない。
52) 出しすぎると後の爆弾処理がきつくなるのだ。窮めた者ならば、いつ点火して爆発するか計算しつつプレイするようだが。
53) 著名なマッドサイエンティスト。世界征服を企む。通称閣下。筆者のお気に入り（^^;
54) 土星には時計がついていた！

三年目に入ると、多くの女の子のときめき度は危険水域に突入してきます。虹野さんはうまくコントロールできたのですが、他の女の子は皆会うたびに頬を赤らめるようになってしまいました。こうなると傷心度が上がりやすくなるので、必然的にデートの回数が増えていきます。まあ、バスケットボールの練習試合で勝ちつづけたらどうしてもそうなりますね。清川さんなんかは、ときめくようになると[55] 下校時に恥ずかしがって一緒に帰ってくれなくなるのですが、こうなると会っただけで他の娘の傷心度が少しづつ上がるので少々厄介です。

この頃には夜もだいぶ更けまして、寝ながら聞いていた土星の持ち主も、いつの間にか別の部屋へ逃げてしまいました[56]。でもその直後の三年目の秋には詩織はぼろぼろ状態になってまして、二週連続でデートに誘われるという快挙を成し遂げてしまいました。実はその裏で優美と虹野さんの爆弾処理が迫っていた[57] のですが、なんとかなってしまいました。土星版の難易度はかなり低いようです。

結局ゲームは主人公のパラメータも最低線を大幅に越えて[58]、春夏秋冬及びクリスマスの特別イベント[59] を見た上で詩織告白エンディングを無事に迎えました。アルバムは 42 枚に達したようです。結局 4 時過ぎまでかかってしまいました。総プレイ時間は 8 時間弱[60] でしょうか。気になる汚染計画の達成度ですが、気がつかぬうちに睡眠学習された者や、画面写真が目に焼き付いてしまった者など、汚染を受けた部員は多数に上ったようです。なかなか愉快な結果ですねえ。今後[61] が楽しみです。ふふふ[62]。

翌朝は 8 時前でも前夜の疲れでくたばっているマグロだらけでした。朝食をとったらすぐに荷物を片付けて一つの部屋にまとめました。で、片付ける前に昨夜まで大活躍したゲーム機達を写真に収めました。そして昼前までオリンピックの野球の決勝戦日本対キューバを見ていました。6–1 から 5 点取って並ぶあたりは圧巻でしたね。結局乱打戦の末、キューバが金メダルを勝ち取りました。昼前に須磨氏が通信をしに出かけるのに付き合って村の郵便局まで行き、これも証拠写真[63] を撮りました。昼食はスパゲティでしたが、少々量が少なかったようです。

一時過ぎにバス[64] に乗って帰路につきます。中野から上信越道に入り、更埴[65] でそのまま長野自動車道に、梓川[66] で小休憩してから岡谷で中央道に入りました。その後は

[55] 確実に運動会で特別イベントが起る。運動会イベントは、ある意味やばい (謎)
[56] 大丈夫。しっかり睡眠学習を受けていたし、データは彼のマシンに残っている。いつでもやりなおせるぞ。
[57] この辺のスリルがこのゲームの醍醐味だったりして。
[58] 正月に神社で学業成就の願をかけると、パラメータがどんと上がる。
[59] 普通はデート先で会話の三択があったりするが、条件を満たすと特別な絵の出るイベントが起るのだ。でも相合い傘は中々見られないな (謎)
[60] ベテランは 3 時間強で解く。
[61] 例の物が発売される 9 月 26 日以降の事を指しているらしい (激ぉ
[62] OB 合宿でも持って行こうかな…
[63] 電話ボックスの中は、直射日光のためにかなり暑かった。
[64] 車内でひろし君と濃い話をしていたのに気がつかなかった人はいるかな？
[65] 上信越道の佐久–更埴間は、まだ開通していなかった。
[66] 松本のすぐ近く。

途中談合坂[67] で休憩し、霧や渋滞に巻き込まれながらも予定通り七時過ぎに新宿西口に帰還しました。焼酎やウイスキーの瓶が残ってしまい、処分に困りましたがとりあえず私が家に持ち帰り、後日根津研に運んだのでした。

この日は結構早く家に帰ることができたのですが、徹メモの代償はあまりにも大きく、翌日は海の方[68] にいく元気が全く残っていなかったのでした。

BGM ときめきメモリアルオリジナルゲームサントラ (これって聴いてるとどうしても笑い転げてしまうのだが、なんでかくも愉快なパロディ曲ばっかり詰まっているのだろう…。曲数もやたらあるし。)

[67] 大月から少し東京寄り。
[68] この日のビッグサイトは煩悩で満ち溢れていたとか。しかしそれなりに涼しかったらしい。

文字コードの話

後編

早坂くりす

4　ISO 2022

この章では、ISO 2022 という規格について解説します。この規格は、複数の文字集合を組み合わせ、切り換えて使うための枠組みを規定したもので、文字コード体系を理解する上でぜひ知っておく必要のある重要な規格です。

なお、ISO 2022 は JIS X 0202 という JIS 規格にもなっています。

4.1　ISO 2022 の考え方

文字コードは一般に 7 ビットまたは 8 ビットのバイト列で構成されます。一方、表現すべき文字のほうは、欧米でも数百字、漢字文化圏では数千〜数万字に達します。しかも使われる文字は言語や国によって大きく異なるため、各国がそれぞれに文字集合を規格化しています。すなわち、コード空間の狭さに比べて文字種ははるかに多く、かつ多数の文字集合が存在しています。

このような状況で、さまざまな文字データを支障なく表現・交換するためには、各国の文字集合の規格を組み合わせ、また切り換えて使うことが考えられます。すなわち、普段必要とする文字を含む文字集合を組み合わせて運用し、それ以外の文字が必要となったときはそれを含む文字集合に動的に切り換えます。この枠組みを規定したのが ISO 2022 です。一方、これに対して、コード空間を大きくとり、その中にあらゆる文字を含めるという方法も考えられます。これを実現したのが ISO 10646/Unicode です。

4.2　7 単位系の指示と呼び出し

それでは、いよいよ ISO 2022 の具体的な解説に入っていきましょう。

まずは 7 単位系 (7 ビット符号) から解説します。7 ビットの符号空間は 0x00 〜0x7F の範囲がありますが、この空間をインユーステーブル (in-use table) といいます。

7 単位系のインユーステーブルは 2 つの領域に分けられます。一つは制御文字を割り当てる領域で、C0 領域といい、0x00〜0x1F です。残りの 0x20〜0x7F は、図形文字を割り当てる領域で、GL 領域といいます。このそれぞれの領域にどの文字集合を割り当てるかが決まれば、各々の符号の表現する文字もめでたく決まるわけです。

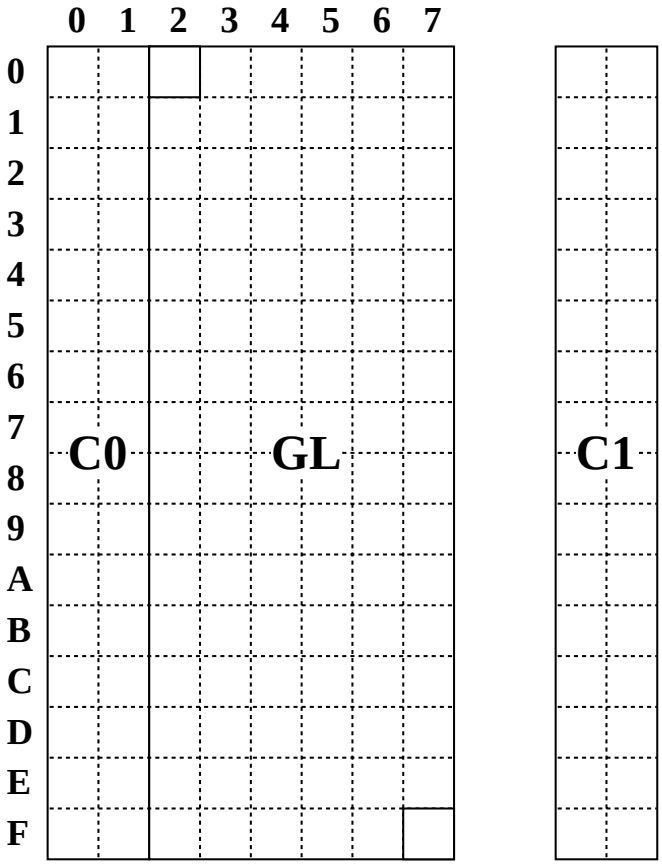

図 1: ISO 2022 のインユーステーブル (7 単位系)

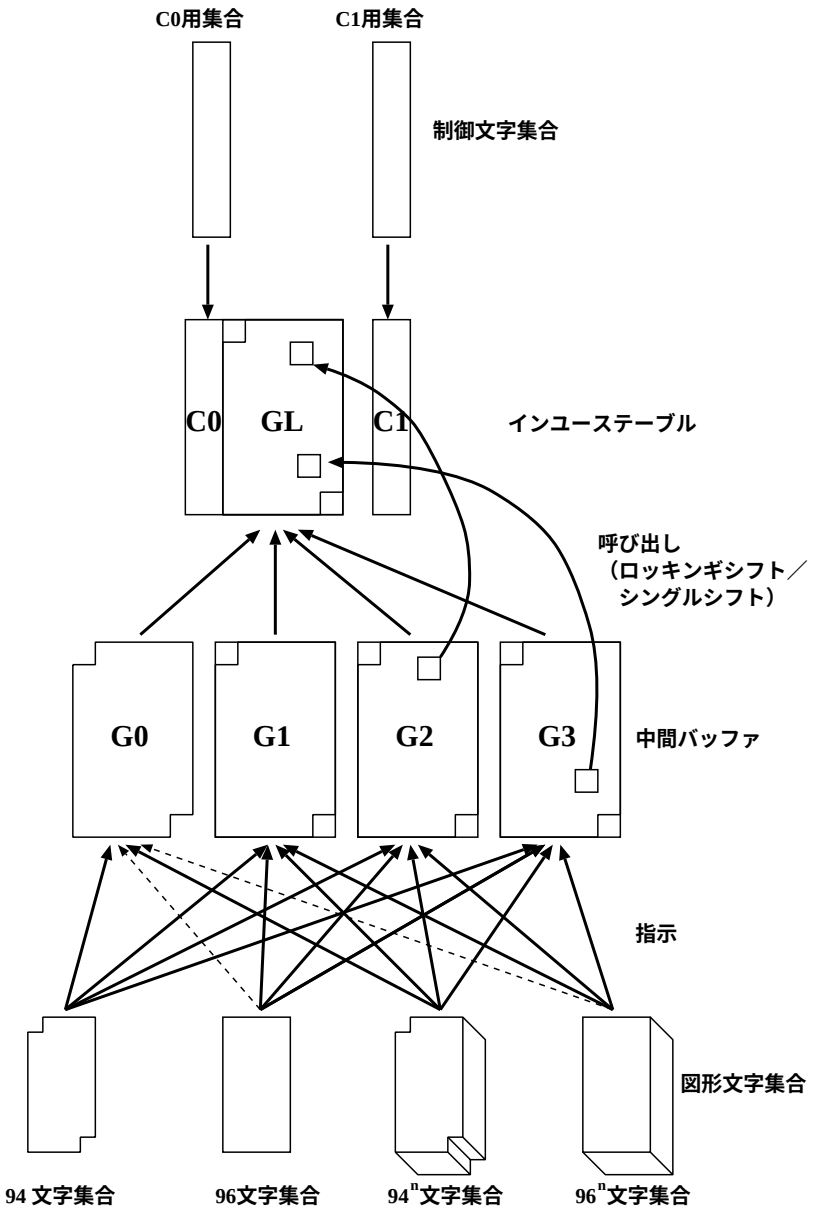

図 2: ISO 2022 の構造 (7 単位系)

文字集合の割り当ての機構は、制御文字と図形文字で異なっています。　制御文字の場合は、単に、所定のエスケープシーケンスによって、C0 領域にどの制御文字集合を割り当てるかを決定します。これに対して、図形文字集合の割り当ては二段構えになっています。インユーステーブルの他に、G0〜G3 という 4 個の中間バッファが設けてあり、図形文字集合をいずれかの中間バッファに割り当て、それからいずれかの中間バッファを GL 領域に割り当てるという機構になっています。図形文字集合を中間バッファに割り当てることを「指示する (to designate)」といい、中間バッファを GL 領域に割り当てることを「呼び出す (to invoke)」といいます。指示はエスケープシーケンスによって、また呼び出しは制御文字によって行われます。

このような複雑な機構になっている理由は後ほど考えることにしますが、この指示/呼び出しの機構が ISO 2022 の中核となる部分ですので、よく理解しておいてください。インユーステーブル・中間バッファ・図形文字集合という階層構造のモデルは、コンピューターの記憶装置におけるレジスタ・主記憶・外部記憶といった階層構造になぞらえることもできます。

なお、指示と呼び出しは独立の機能ですので、指示した後すぐに呼び出さなくてはならないということはありません。また、すでにインユーステーブルに呼び出されている中間バッファに対して図形文字集合を指示すると、改めて呼び出しをしなおさなくても、即座に指示がインユーステーブルに反映されることになっています。

4.3　ISO 2022 で使える文字集合

ここでは、ISO 2022 で使える文字集合について説明します。ISO 2022 はどんな文字集合でも使えるわけではなく、上述のインユーステーブルの構造と一致する文字集合でなければなりません。制御文字については、C0 領域が 0x00〜0x1F の 32 文字なので、32 文字の制御文字集合を用いることができます。図形文字は、GL 領域が 0x20〜0x7F の 96 文字なので、96 文字の図形文字集合を用いることができます。

ただし実際には、ASCII/ISO 646 との互換性のために、GL 領域を 0x21〜0x7E の 94 文字に制限して、0x20 は空白文字、0x7F は制御文字 DEL (削除) として使われるのが普通です。この場合、94 文字の図形文字集合を用いることができます。94 文字集合を GL 領域に呼び出すと、自動的に 0x20 は空白文字、0x7F は DEL になります。

94 文字図形文字集合の例としては、ASCII 図形文字、ISO 646 図形文字 (JIS X 0201 ローマ文字を含む)、JIS X 0201 カタカナなどがあります。96 文字図形文字集合の例としては、ISO 8859 の右半分などがあります。

漢字のような文字数の多い図形文字は、94 文字/96 文字集合では収容できませんので、複数バイト文字集合として規定されます。すなわち、

2 バイト符号の場合は、94 × 94 文字または 96 × 96 文字
3 バイト符号の場合は、94 × 94 × 94 文字または 96 × 96 × 96 文字

となります。例えば、JIS X 0208 は 94 × 94 文字図形文字集合であり、1 区 1 点から 94 区 94 点までの 8836 個の符号空間を持ちます。ちなみに、古い ISO 2022 では、複数バイト文字集合は G0 にのみ指示できることになっていましたが、現在ではこの制限はありません。

なお、96 文字集合や 96^n 文字集合を G0 に指示することはできないことになっています。G0 は GL に呼び出されるべきデフォルトのバッファであると考えられるため、ここに 96 文字集合が指示されていると、0x20 が空白でなくなるなど問題が多いからかもしれません (GL に 96 文字集合を呼び出すこと自体は、G1〜G3 を使えば可能です)。

4.4 呼び出し: ロッキングシフトとシングルシフト

上述のように、図形文字集合用の中間バッファG0〜G3 を GL 領域に割り当てることを「呼び出す」といいますが、これには 2 通りの方法があります。一つはロッキングシフトといい、いったん割り当てたら次に割り当て直すまでずっと有効となるものです。もう一つはシングルシフトといい、1 文字分だけ臨時に呼び出しを変更するものです。呼び出しのパターンと、それを実現する制御文字の名称を以下に示します。

呼び出しの方法	呼び出しの内容	制御文字
ロッキングシフト	G0 → GL	`SI` (Shift In)
	G1 → GL	`SO` (Shift Out)
	G2 → GL	`LS2` (Locking Shift 2)
	G3 → GL	`LS3` (Locking Shift 3)
シングルシフト	G2 → GL	`SS2` (Single Shift 2)
	G3 → GL	`SS3` (Single Shift 3)

G0/G1 のロッキングシフトは `LS0`/`LS1` でなく `SI`/`SO` という名称になっています。また、シングルシフトは G2/G3 にしかありません。

シングルシフトの制御文字 `SS2` (`SS3`) が現れたら、それに続く 1 文字分の符号を、G2 (G3) に指示されている図形文字集合の符号として解釈します。すなわち、後続の 1 文字分の符号に限って臨時に呼び出しを変更することに相当します。なお、「1 文字分の符号」のバイト数は、G2 (G3) に指示されている図形文字集合によって決まります。必ずしも 1 バイトとは限りません。

4.5 8 単位系の指示と呼び出し

ISO 2022 では 8 単位系 (8 ビット符号) についても規定されています。8 単位系のインユーステーブルの構造は、7 単位系との互換のために、7 単位系のそれを二つ組み合わせたような形になっています。

C0 領域	0x00〜0x1F	制御文字の領域
GL 領域	0x20〜0x7F	図形文字の領域
C1 領域	0x80〜0x9F	制御文字の領域
GR 領域	0xA0〜0xFF	図形文字の領域

指示および呼び出しの機構は基本的に 7 単位系と同じです。

制御文字の領域は C0/C1 の 2 つありますが、制御文字集合はそれぞれ C0 用または C1 用のいずれかと決められており、C0 用制御文字集合を C1 に割り当てることやその逆はできません。図形文字はそのような制限はなく、基本的には自由に指示し呼び出すことができます (ただし、7 単位系のところで述べたように、いくつか例外的な制約があります)。

中間バッファ G0〜G3 は 7 単位系と同じですが、インユーステーブルに GR 領域が追加されたことに伴い、呼び出しの機能が追加・変更されています。

呼び出しの方法	呼び出しの内容	制御文字
ロッキングシフト (左)	G0 → GL	`LS0` (Locking Shift 0)
	G1 → GL	`LS1` (Locking Shift 1)
	G2 → GL	`LS2` (Locking Shift 2)
	G3 → GL	`LS3` (Locking Shift 3)
ロッキングシフト (右)	G1 → GR	`LS1R` (Locking Shift 1 Right)
	G2 → GR	`LS2R` (Locking Shift 2 Right)
	G3 → GR	`LS3R` (Locking Shift 3 Right)
シングルシフト	G2 → GL/GR	`SS2` (Single Shift 2)
	G3 → GL/GR	`SS3` (Single Shift 3)

7 単位系と大きく異なるのは、GR への呼び出しを行う `LS?R` が追加されたことです (ただし、G0 を GR に呼び出すことはできないことになっています)。また、7 単位系の `SI`/`SO` が `LS0`/`LS1` とわかりやすい名前に改められました。シングルシフトは GL に呼び出すのが原則ですが、GR に呼び出してもよい (シングルシフトに続く符号の MSB が 1 でもよい) ことになっています。

複数バイト図形文字集合も GR または GL のいずれか一方に呼び出されますので、2 バイト文字の第 1 バイトを GR に割り当てて第 2 バイトを GL に割り当てるといったようなことはできません。7 単位系との連続性を優先した結果ですが、符号空間の有効利用という観点からは不満の残るところです。

なお、94 文字集合や 94^n 文字集合を GL に呼び出した場合は、0x20 が空白で 0x7F が `DEL` になりますが、GR に呼び出した場合は、0xA0 と 0xFF は使用禁止となります。

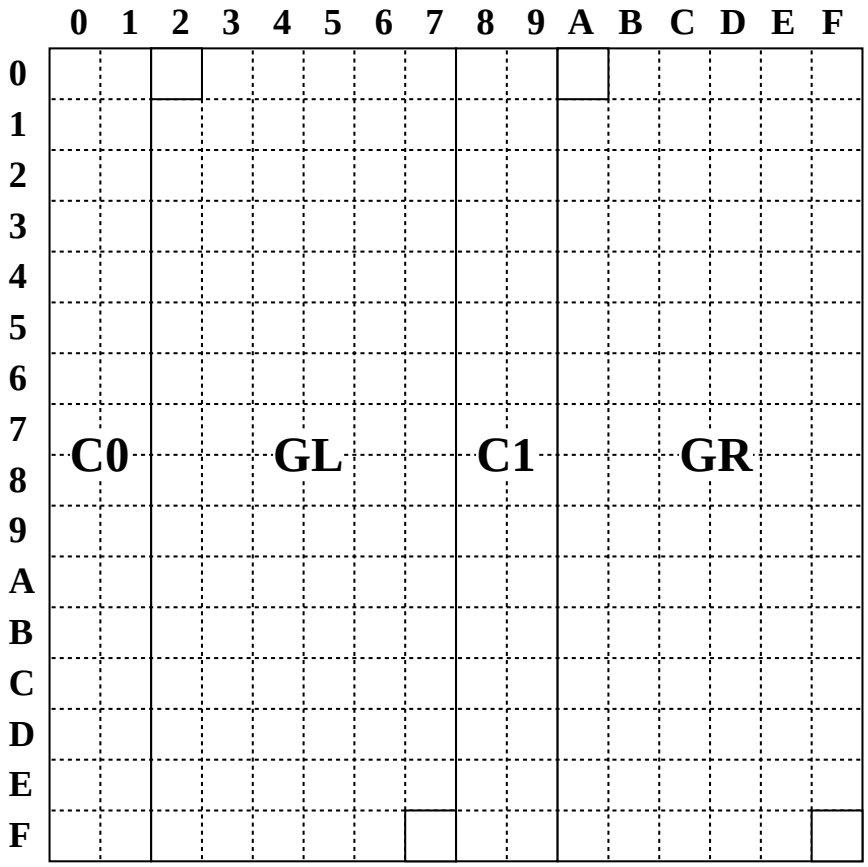

図 3: ISO 2022 のインユーステーブル (8 単位系)

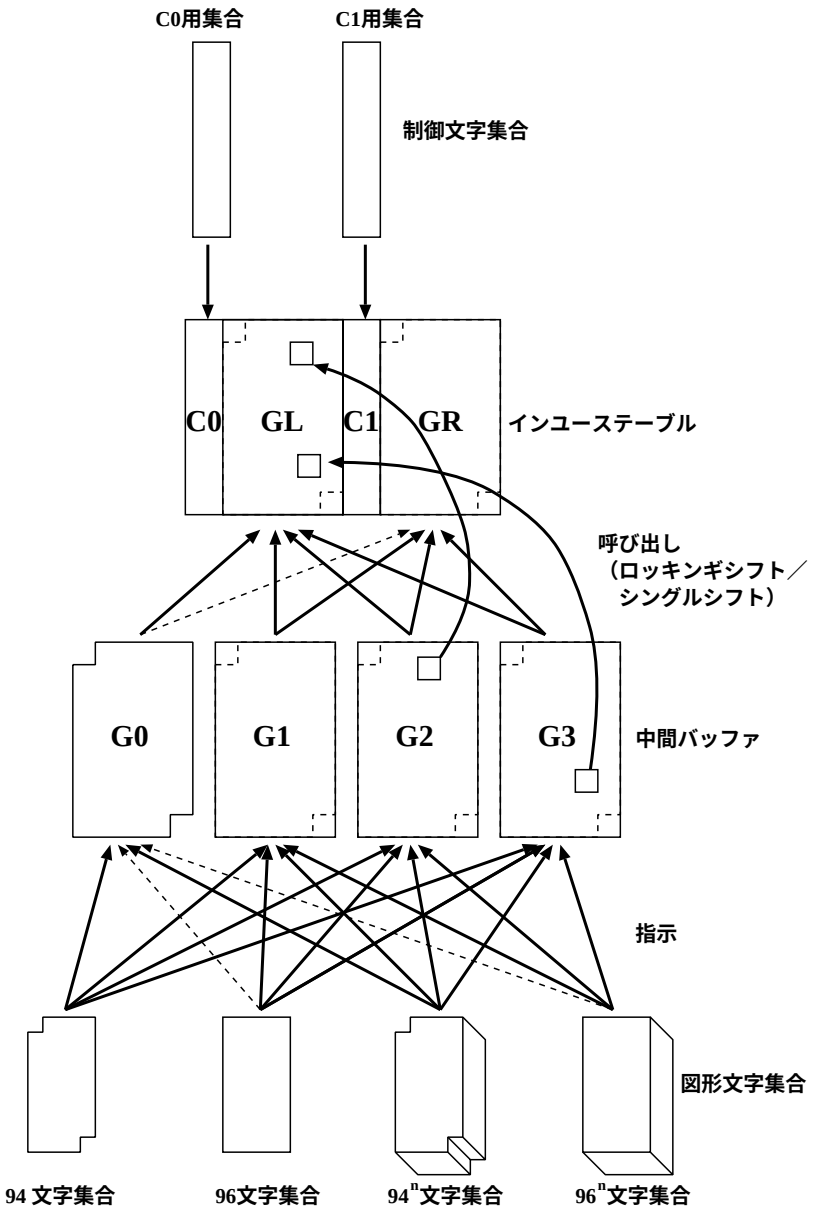

図 4: ISO 2022 の構造 (8 単位系)

4.6 ISO 2022 の制御機能

これまでみてきたように、ISO 2022 では指示や呼び出し (ロッキングシフト、シングルシフト) といった制御機能をもっていますが、これは実際には制御文字やエスケープシーケンスを用いて表現されます。制御文字は C0 制御文字と C1 制御文字があります。また、C0 制御文字の ESC (0x1B) に続いて 0x20〜0x7E の範囲の符号を並べたものをエスケープシーケンスといいます。ISO 2022 ではこれらを用いて以下のような制御機能を表現します。

- 指示、呼び出し
- 単独の制御機能
- 7 単位系における C1 制御文字
- アナウンサ
- 私用の制御機能

ここではこのうち主なものを紹介します。

ISO 2022 のエスケープシーケンス

ここでは ISO 2022 のエスケープシーケンスの一般形について述べます。ISO 2022 で規定されるエスケープシーケンスは、以下のような形をしています。

〈ESC〉 〈0 個以上の中間文字...〉 〈終端文字〉

中間文字とは 0x20〜0x2F の範囲の符号であり、終端文字とは 0x30〜0x7E の範囲の符号です。

なお、エスケープシーケンスを構成する中間文字や終端文字は単なる符号であり、特定の文字集合の文字を表現するものではありませんが、便宜上 ESC $ B のように ASCII 文字を用いて示すことがあります。

図形文字集合の指示

文字集合の指示はエスケープシーケンスによって表現されます。終端文字 *Ft* は指示すべき文字集合を表します。

ESC (Ft	94 文字集合を G0 に指示
ESC) Ft	94 文字集合を G1 に指示
ESC * Ft	94 文字集合を G2 に指示
ESC + Ft	94 文字集合を G3 に指示
ESC - Ft	96 文字集合を G1 に指示
ESC . Ft	96 文字集合を G2 に指示
ESC / Ft	96 文字集合を G3 に指示
ESC $ Ft	94^n 文字集合を G0 に指示 (注)
ESC $ (Ft	94^n 文字集合を G0 に指示
ESC $) Ft	94^n 文字集合を G1 に指示
ESC $ * Ft	94^n 文字集合を G2 に指示
ESC $ + Ft	94^n 文字集合を G3 に指示
ESC $ - Ft	96^n 文字集合を G1 に指示
ESC $. Ft	96^n 文字集合を G2 に指示
ESC $ / Ft	96^n 文字集合を G3 に指示

すでに述べたとおり、96 文字集合や 96^n 文字集合を G0 に指示することはできないことになっています。

(注) かつては複数バイト文字集合は G0 にしか指示できなかったので、この形のエスケープシーケンスが使われていました。この制限は後に撤廃され、下に示したような一般化された形のエスケープシーケンスに改められました。ただし、この時点で登録されていた 94^n 文字集合に限っては、互換性のためにこの形のエスケープシーケンスを引き続き用いてもよいことになっています。

終端文字と文字集合の対応は登録制になっており、ECMA という組織が登録簿を管理しています。以下にいくつか例を示します。

- 94 文字集合
  - @ ISO 646 IRV
  - B US ASCII
  - H スウェーデン名前用文字
  - I JIS X 0201 カタカナ
  - J JIS X 0201 ローマ文字
- 96 文字集合
  - A ISO 8859-1 右半分

- 94^n 文字集合
  - @ JIS X 0208-1978 (旧 JIS 漢字)
  - A GB 2312-80 (中国語簡体字)
  - B JIS X 0208-1983 (新 JIS 漢字)
  - C KS C 5601-1987 (韓国語ハングル・漢字)
  - D JIS X 0212-1990 (補助漢字)

文字集合の区分は中間文字によって識別されるので、終端文字が同一でも中間文字が異なれば異なる文字集合を表すことに注意してください。

JIS コードで使用される指示のシーケンスは以下のようになります。

ESC (B　G0 に ASCII を指示する
ESC (J　G0 に JIS X 0201 ローマ文字を指示する
ESC $ @　G0 に JIS X 0208-1978 を指示する
ESC $ B　G0 に JIS X 0208-1983 を指示する

呼び出しの制御機能など

呼び出しの制御機能の表現は、C0 制御文字・C1 制御文字・単独の制御機能と多岐にわたっています。単独の制御機能とは、制御文字集合に含まれない制御機能をエスケープシーケンス ESC 0x60〜0x7E で表現するものです。

制御機能	符号	符号の区分
SI/LS0	0x0F	C0 制御文字
SO/LS1	0x0E	C0 制御文字
LS2	ESC 0x6E	単独の制御機能
LS3	ESC 0x6F	単独の制御機能
LS1R	ESC 0x7E	単独の制御機能
LS2R	ESC 0x7D	単独の制御機能
LS3R	ESC 0x7C	単独の制御機能
SS2	0x8E	C1 制御文字
SS3	0x8F	C1 制御文字

ところで、 SS2/SS3 が C1 制御文字で表現されていますが、これでは 7 単位系で SS2/SS3 が表現できませんね。実は、7 単位系でも C1 制御文字を表現する方法があるのです。7 単位系で C1 制御文字を表現するには、C1 領域 0x80〜0x9F の代わりにエスケープシーケンス ESC 0x40〜0x5F を使うことになっています。したがって、7 単位系の SS2/SS3 は以下のようになります。

SS2　ESC 0x4E　C1 制御文字
SS3　ESC 0x4F　C1 制御文字

4.7　ISO 2022 の運用

以上、ISO 2022 の主要部分をみてきました。これだけでも ISO 2022 が非常に複雑で多様な規格であることがわかってもらえたと思います。このような複雑な規格のフルセットを実装・運用することは現実的てはありません。むしろ、ニーズに応じて適切なサブセットを定めて運用するのが現実的でしょう。

指示や呼び出しは、情報交換当事者同士の合意があれば省略できることになっています。たとえば、多くとも 4 個の文字集合で足りるようなシステムでは、使用する文字集合をあらかじめ G0〜G3 に指示してあることにすれば、呼び出しだけで文字集合を切り換えることができます。8 単位系で文字集合が 2 個までであれば、呼び出しすら必要ありません。一方、多くの文字集合を必要とするシステムでは、呼び出しを固定して指示を切り換えるのが適切でしょう。

このように、とくに図形文字集合の扱いにおいてさまざまなサブセットを構成できるようになっているため、全体としてはおおげさとも思えるほどの規模の規格になっているといえます。

5　ISO 2022 の実例

この章では、ISO 2022 の実例について解説します。

5.1　EUC

EUC は ISO 2022 に準拠したエンコーディング法です。日本語 EUC を例にとって説明します。

G0 に ASCII(または JIS X 0201 ローマ文字) を指示
G1 に JIS X 0208 漢字を指示
G2 に JIS X 0201 カタカナを指示
G3 に JIS X 0212 補助漢字を指示
G0 を GL に、G1 を GR に呼び出す
G2 と G3 はシングルシフトで使用
エスケープシーケンス・ロッキングシフトは使わない

EUC では X 0208 は第 1 バイトと第 2 バイトの MSB が 1 になりますが、これは単に ASCII と区別するために適当にそうしたのではなく、ISO 2022 の枠組みに沿って符号化した結果というわけです。カタカナや補助漢字に付く 0x8E/0x8F のプレフィクスも、ISO 2022 のシングルシフトに由来しています。

EUC の一般的な構成として、ASCII (または ISO 646) を含めて 4 つまでの文字集合を扱うことができます。これらは G0〜G3 に指示されています。特に、G0 には ASCII (ISO 646)、G1 にはその言語でもっともよく使う文字集合 (ASCII 以外) が指示されており、これらはそれぞれ GL と GR に呼び出されています。指示および呼び出しは固定されており、エスケープシーケンスやロッキングシフトで変更されることはありません。さらに、補助的に用いる文字集合が G2 と G3 に指示されており、これらはシングルシフトによって呼び出されます。指示・呼び出しが固定なので、文字列を解釈または生成する際、現在どの文字集合が指示されているかといったような「状態」情報を保持する必要がない (stateless, modeless) ことが大きな特徴です。

なお、通常 EUC と呼んでいるものは、正確には情報交換用の EUC packed format という形式で、文字集合によって 1 文字のバイト数が変化します。EUC には内部処理用の EUC 2-byte complete format と呼ばれる形式もあり、これは 1 バイト文字も 2 バイト文字もすべて 2 バイトで表現し、ISO 2022 非準拠です。

5.2 JIS エンコーディング (JUNET コード、ISO-2022-JP)

JIS (漢字) コードという俗称は、次の二つの意味で使われています。

1. JIS X 0208 を GL に呼び出した状態でのビット組み合わせ。
2. 1. と ASCII などと共存させるために、ISO 2022 準拠の指示のエスケープシーケンスなどを用いるエンコーディング法。

2. は第 1 章で説明した JIS コードです。今後は 2. を JIS エンコーディングと呼ぶことにします。これは典型的な ISO 2022 準拠のエンコーディング法です。

> G0 に JIS X 0201 ローマ文字 (または ASCII) を指示
> G1 に JIS X 0201 カタカナを指示
> G0 を GL に呼び出す
> JIS X 0208 漢字 (および他の文字集合) は G0 に指示して使用
> (場合によっては JIS X 0201 カタカナも G0 に指示)
> 7 ビット環境では G1 は GL に呼び出して使用することが多い
> シングルシフトは使わない

とくに、JIS エンコーディングの 7 ビット版は、日本のインターネットの母体となった JUNET というネットワークにおいて、日本語メッセージ (メール、ニュース) の交換の

ために使われ、現在に至っています。そのため、時に JUNET コード (Mule の `*junet*`) とも呼ばれます。また、ISO-2022-JP という名前で RFC 1468 に定義されています。

JIS エンコーディングの特長は、7 ビット環境でも使えることだけでなく、サポートする文字集合をいくらでも増やせること (使いたい集合を G0 に指示するだけでよい) です。ISO-2022-JP では ASCII・JIS Roman・X 0208 (旧/新) だけに限定していますが、これの拡張である ISO-2022-JP-2 (RFC 1554) では、カタカナ・補助漢字や中国・韓国のコードも追加されています。また、ISO 8859 も G2 に指示して使えることになっています。

このエンコーディング法を「JIS」と呼ぶのは、いくばくかの違和感を禁じえません。というのも、たしかに JIS X 0202 (ISO 2022) から構成可能ですが、このエンコーディング法そのものが JIS 規格として規定されているわけではないからです。JIS エンコーディングとは「必要とする文字集合を随時 G0 に指示する」という原則からなる不文律のようなものといっていいでしょう。

5.3 シフト JIS

シフト JIS は言うまでもなく ISO 2022 非準拠です。シフト JIS の欠点は、すべてこの点に集約されます。第 1 章でいくつか具体的な欠点をあげましたが、それらはすべて「ISO 2022 に準拠していない」という致命的な欠点の系にすぎません。たしかにシフト JIS は巧妙に設計されており、日本語処理の黎明期においてカタカナから漢字への円滑な移行を促した役割は評価すべきですが、アドホックな業界標準としての拡張性のなさはいかんともしがたく、今後の国際化・多言語処理の流れに対応できるものではありません。

5.4 ISO 8859、JIS X 0201 など

ISO 8859 や JIS X 0201 などは、それだけで完結した 8 ビットコードの規格ですが、ASCII (ISO 646) ともう一つの追加文字集合を ISO 2022 によって組み合わせたものとみることもできます。X 0201 については上記の JIS エンコーディングの説明でカバーしましたが、ISO 8859 については次のようになります。

G0 に ASCII を指示
G1 に ISO 8859-x 右半分を指示
G0 を GL に呼び出す
G1 を GR に呼び出す
エスケープシーケンス・シフト機能は使わない

これを拡張し、G0/G1 に各種文字集合を呼び出せるようにした多言語エンコーディング法が、X11R5 の国際化機能で採用されました。Compound Text (Mule の`*ctext*`) と

呼ばれます。

5.5 C1 制御文字

C0 の制御文字集合はいくつかありますが、普通に使われているのは ISO 646 (ASCII) のそれでしょう。これはいまさら説明するまでもなく、みなさんがいつも使っているものです。それに比べて、C1 の制御文字というのはなじみが薄い気がします。SS2 (0x8E) と SS3 (0x8F) は ISO 2022 で出てきましたが、これ以外で広く使われているものがあるのでしょうか?

実は、あるのです。MS-DOS や xterm で「画面制御用 ANSI エスケープシーケンス」と呼ばれているものです。これは、ISO 6429 (ANSI X3.64, JIS X 0211) に規定されている C1 集合に準拠しています。ISO 2022 の説明で、「C1 集合の制御文字 (0x80〜0x9F) を ESC 0x40〜0x5F で表現することができる」というのがありましたね。ANSI エスケープシーケンスというのは、ANSI X3.64 の C1 集合をエスケープシーケンスで表現したものだったのです。

例えば、この規格の 0x85 に、復帰改行を行う NEL (next line) という制御文字があります。これをエスケープシーケンスで表現すると ESC 0x45、つまり ESC E となります。DOS のマニュアルなどで ESC E を確認してみてください。

また、同じく 0x9B には CSI (control sequence introducer) というのがあります。これは ESC 0x5B、つまり ESC [となります。ESC [といえば、画面消去・カーソル移動・属性変更など、さまざまなエスケープシーケンスのプレフィクスですね。CSI で始まるこれらのシーケンスは、制御シーケンス (control sequence) と呼ばれています。制御シーケンスは終端文字によってさまざまな機能があり、パラメータをとるものもありますが、これらはすべて ISO 6429 で規定されています。

6 中国語・韓国語の文字コード

この章では、日本語以外に 2 バイト文字を用いる、中国語・韓国語の文字コード体系について解説します。

6.1 GB 2312

中国本土の国家規格は GB (国家標準) といい、2 バイト文字コードの規格も簡体字・繁体字それぞれにいくつか定められています。そのうち最もよく使われるのが、GB 2312

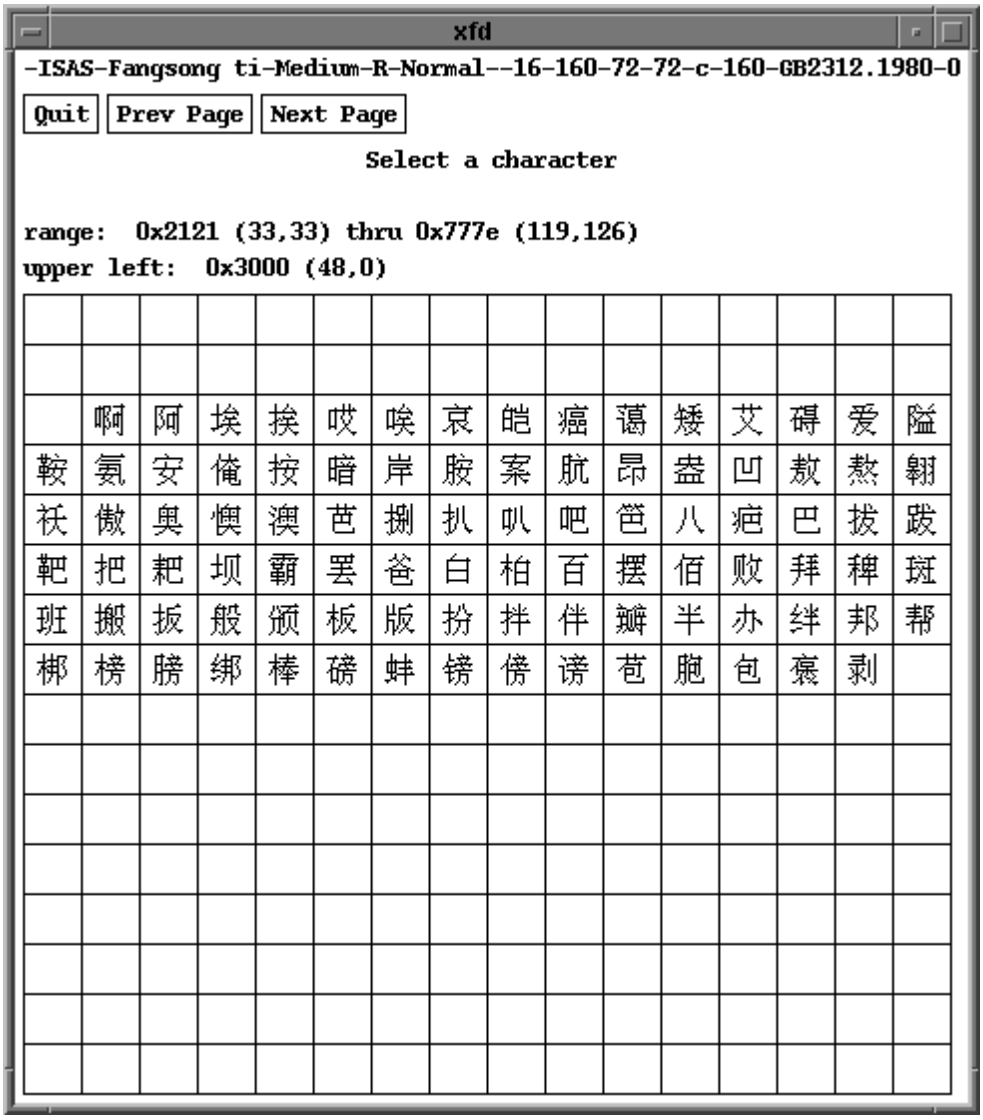
xfd
-ISAS-Fangsong ti-Medium-R-Normal--16-160-72-72-c-160-GB2312.1980-0
Quit
Prev Page
Next Page
Select a character
range: 0x2121 (33,33) thru 0x777e (119,126)
upper left: 0x3000 (48,0)
啊 阿 埃 挨 哎 唉 哀 皑 癌 蔼 矮 艾 碍 爱 隘
鞍 氨 安 俺 按 暗 岸 胺 案 肮 昂 盎 凹 敖 熬 翱
袄 傲 奥 懊 澳 芭 捌 扒 叭 吧 笆 八 疤 巴 拔 跋
靶 把 耙 坝 霸 罢 爸 白 柏 百 摆 佰 败 拜 稗 斑
班 搬 扳 般 颁 板 版 扮 拌 伴 瓣 半 办 绊 邦 帮
梆 榜 膀 绑 棒 磅 蚌 镑 傍 谤 苞 胞 包 褒 剥

図 5: GB 2312

という簡体字の規格です。構成は JIS X 0208 とよく似ており、次のような特徴をもちます。

- ISO 2022 準拠の 2 バイトコード
- 漢字だけでなく英数字や仮名を含む
- 漢字は第 1 級漢字 (読みの順) と第 2 級漢字 (部首順) に分かれる

実際には GB 2312 は GR に呼び出されて EUC として使われるケースが多いようです。ちょうど日本語 EUC の G0・G1 と同じような形になります。

ただし、インターネットでは 8 ビットデータの通過が必ずしも保証されませんので、alt.chinese.text では Hz (Hanzi=漢字) とよばれる独自の 7 ビットエンコーディングを用いています。Hz では、GB 2312 漢字の開始シーケンスとして `~{` を、ASCII の開始シーケンスとして `~}` を使います。

6.2 Big-5 と CNS 11643

台湾の有力メーカー 5 社が中心となって策定したのが Big-5 と呼ばれるコードです。メーカーによって独自拡張がありますが、基本的な部分は、第 1 バイトが 0xA1〜0xF9、第 2 バイトが 0x40〜0x7E、0xA1〜0xFE となっており、設計思想はシフト JIS と似ています (もちろん ISO 2022 非準拠)。ただしコード空間は広くなっており、約 13,000 字もの文字を収容しています (漢字は繁体字で総画数順)。

シフト JIS と本質的に違うのは、シフト JIS が JIS X 0208 を元にしたエンコーディング法であるのに対して、Big-5 は文字集合とエンコーディングが一体となっている点、および、C1 制御文字のコードを避けているので ISO 2022 との親和性が比較的良い点です。Big-5 は台湾や香港の PC で広く使われています。また、alt.chinese.text.big5 など一部のニュースグループでも使われています。

逆に、Big-5 をもとに台湾の国家規格が作られました。CNS 11643 といいます。これは ISO 2022 準拠の 2 バイトコード 7 個からなる膨大な規模の規格です。それぞれを「字面」と呼び、第 1・第 2 字面に Big-5 の文字を収容し、第 3〜第 7 字面はそれ以外の特殊な漢字を収容しています。第 1・第 2 字面の配列は基本的に Big-5 を踏襲していますが、Big-5 の漢字配列の誤りを正しているため、順序は微妙に違います。また、後ろの方の字面になると、まるで書き間違いのような面妖な字が延々と並んで奇観を呈しています。CNS 11643 は政府機関などを中心に使われているそうです。ISO 2022 準拠のため、EUC を構成することができ、ワークステーションなどへの採用も進んでいると聞きます。

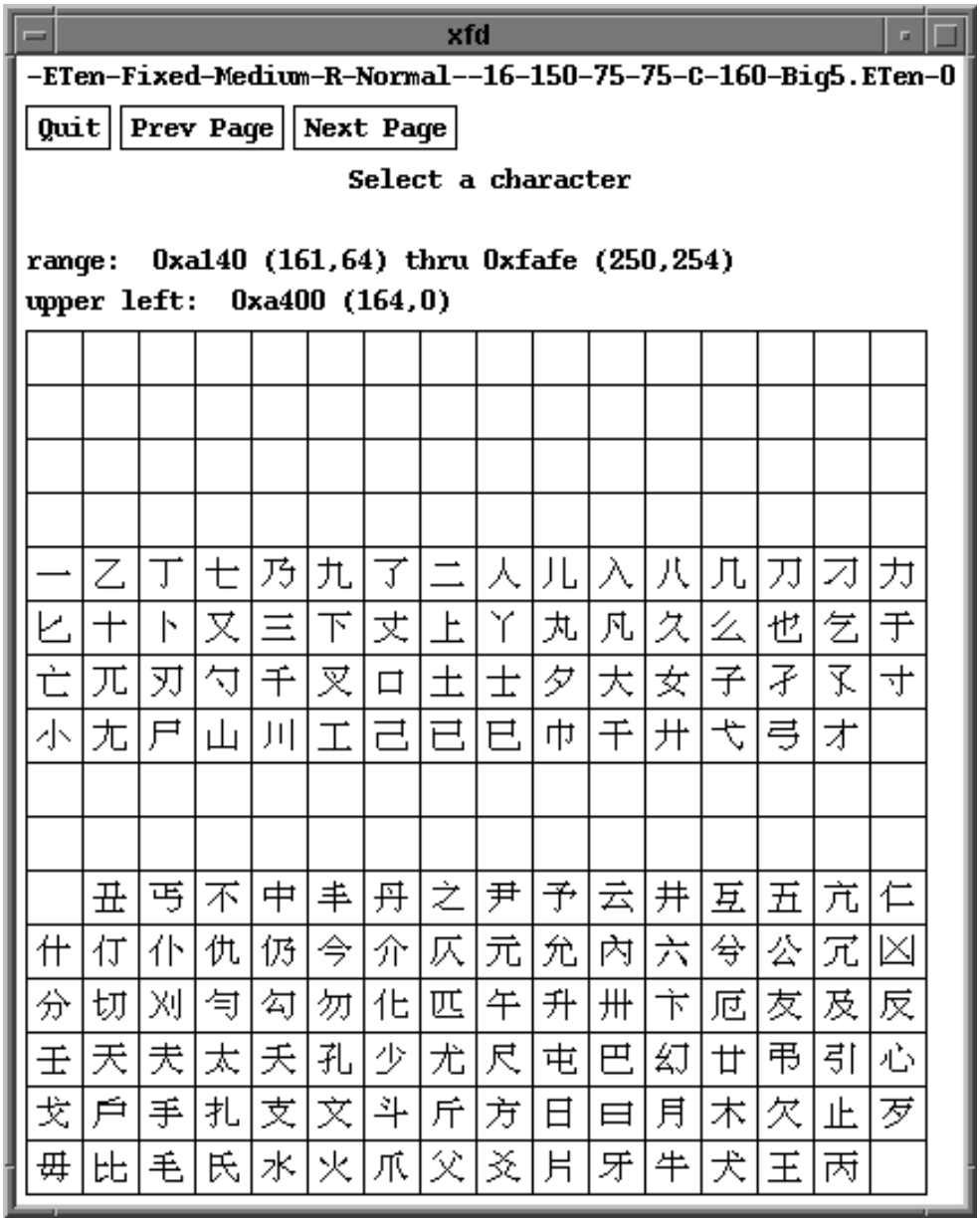
xfd
-ETen-Fixed-Medium-R-Normal--16-150-75-75-C-160-Big5.ETen-0
Quit
Prev Page
Next Page
Select a character
range: 0xa140 (161,64) thru 0xfafe (250,254)
upper left: 0xa400 (164,0)

図 6: Big-5

6.3 KS C 5601

韓国にも JIS や GB 同様、ISO 2022 準拠の 2 バイトコードの規格があります。そのうち、JIS X 0208 や GB 2312 のように最もよく使われるのが KS C 5601 です。これは英数字や仮名の他、ハングル 2350 字、漢字 (繁体字) 約 4000 字が含まれています。

ハングルという文字は、子音字母と母音字母の組み合わせによって 1 文字を構成します。具体的には、初声 (子音 19 種)+中声 (母音 21 種)、または初声+中声+終声 (子音 27 種) という構造になっていて、計算上は $19 \times 21 \times (27 + 1) = 11172$ 字存在します。KS C 5601 には、そのうちよく使われるものだけが入っています。また、漢字は読みの順に並んでいます。一部の漢字は複数の読みを持ちますが、これらがそれぞれの読みの所に重複して収録されているのが大きな特徴です。

KS C 5601 は、通常は GR に呼び出して韓国語 EUC の形で使います。ただし、インターネットでは 7 ビットでなくてはいけませんので、ISO-2022-KR という 7 ビットのエンコーディング法 (ISO-2022-JP にちょっと近い) で使います。

6.4 組合せ型ハングルコード

ハングルの符号化法には 2 通りあります。1 つは、KS C 5601 のように、子音と母音を組み合わせてできた形に対して符号を割り当てるもので、完成型といいます。一方、ハングルの構成原理にのっとって、1 文字分のビット (通常 16 ビット) を 3 つの部分にわけ、それぞれが初声・中声・終声を表すという方法があります。これは組合せ型といいます。

組合せ型コードは MS-DOS で使われていて、実装したメーカーにより何種類かありますが、ここでは KS C 5601 の付録に収録されたものについて解説します。初声・中声・終声はそれぞれ 5 ビットで表せるので、これに識別用の 1 ビットを加えて 16 ビットとします。つまり、バイトの MSB が 1 であればハングルコードとみなし、次のバイトと合わせて 15 ビットでハングル 1 字を表します。中声の 5 ビットがバイト境界にまたがっていたりするので、第 2 バイトにいろいろとまずいコードが現れそうな気がしますが、これが実に巧妙にできていて、C0/C1 制御文字や空白のコードは現れないようになっています。さすがに GL の図形文字のコードと重なるのは避けられず、そのため ISO 2022 にも非準拠ですが、Big-5 同様 C1 のコードを避けているので ISO 2022 との親和性はよくなっています。

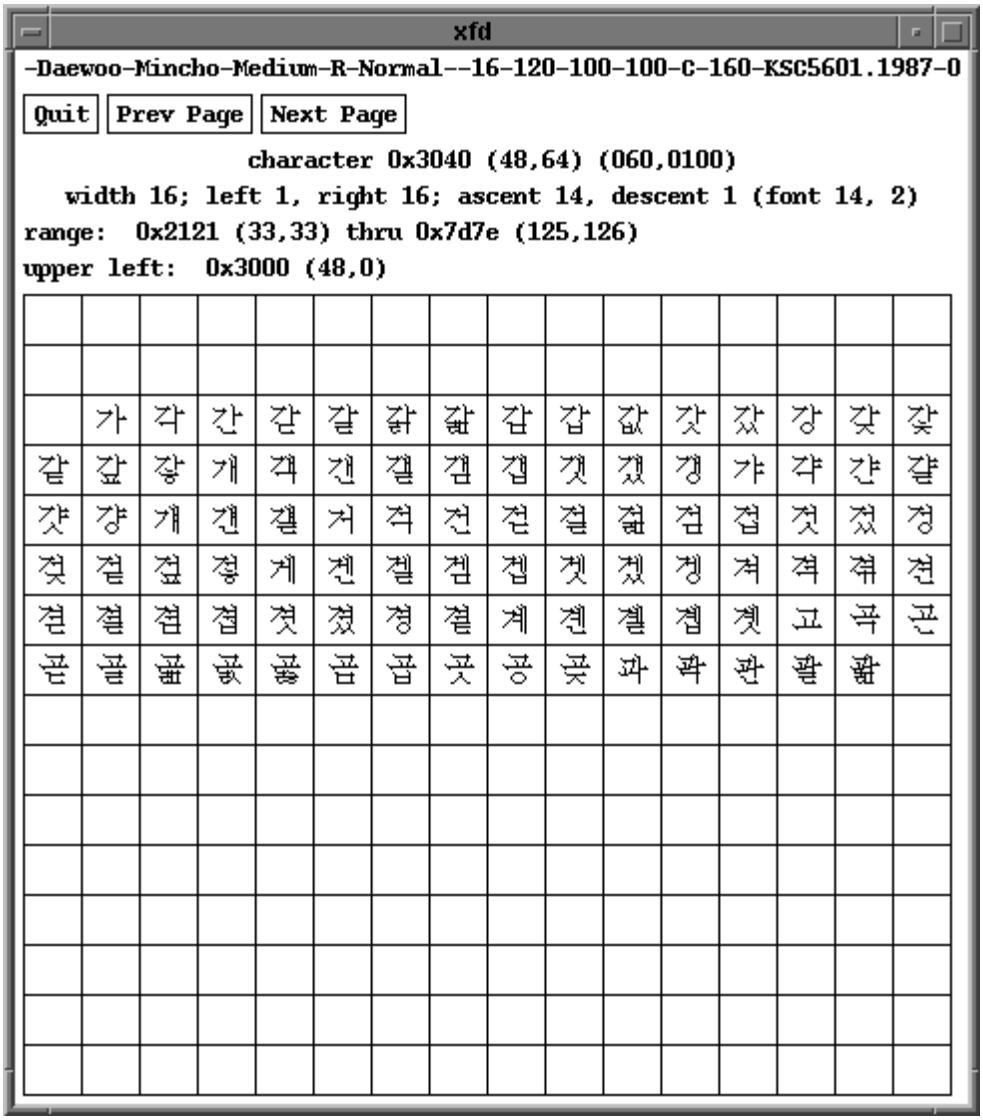
xfd
-Daewoo-Mincho-Medium-R-Normal--16-120-100-100-C-160-KSC5601.1987-0
Quit
Prev Page
Next Page
character 0x3040 (48,64) (060,0100)
width 16; left 1, right 16; ascent 14, descent 1 (font 14, 2)
range: 0x2121 (33,33) thru 0x7d7e (125,126)
upper left: 0x3000 (48,0)

図 7: KS C 5601

7 ISO 10646 と Unicode

この章は、最近何かと話題の ISO 10646 と Unicode の話です。

7.1 ISO 10646 成立の経緯

ISO では、全世界の主要な文字を含んだ単一の文字集合 ISO 10646 の策定作業に着手しました。一方、国際市場を重視し始めたアメリカの有力コンピュータ企業は、同様の目的で、しかし 10646 とは全く異なる Unicode (Ver.1.0) という文字集合を策定しました。

ISO 10646 は規格原案 (DIS 10646 第 1 版) が投票にかけられましたが、10646 と Unicode の二者並立を嫌う意見により否決されました。そこで、Unicode の修正案 (Ver.1.1) が DIS 10646 第 2 版として投票にかけられ、ISO 10646-1 として成立しました。

その後、ISO 10646-1 は JIS X 0221 として JIS 規格にもなりました。

7.2 UCS の構造と特徴

ISO 10646 は UCS (Universal Character Set, 万国文字集合) と名付けられています。UCS は 1 文字 2 バイトの形式 (UCS-2) と 4 バイト (31 ビット) の形式 (UCS-4) があります。UCS-4 の 4 バイトの値はそれぞれ群 (group)、面 (plane)、区 (row)、点 (cell) と呼ばれます。256 の点と 256 の区で表される 65536 個のコードが 1 つの面を構成し、256 の面で 1 つの群を構成し、128 の群で UCS-4 の全体を構成します (群の MSB は未使用)。UCS-4 の最初の面、つまり群 00 の面 00 を BMP (Basic Multilingual Plane、基本多言語面) と呼びます。現在、BMP 以外の面には文字は定義されていません。また、群 00 面 0xE0〜0xFF と群 0x60〜0x7F は私用のために予約されています。

BMP のコードを 2 バイトで表現したものが UCS-2 であり、Unicode はこれとほぼ同じものです。

BMP は以下のように大別されます。

A-zone	0x0000〜0x4DFF
	漢字以外の文字・記号
	制御文字、アルファベット、各種記号、ハングル、仮名など
I-zone	0x4E00〜0x9FFF
	漢字
O-zone	0xA000〜0xDFFF
	拡張用予約領域
R-zone	0xE000〜0xFFFD
	私用予約領域、互換用文字など

A-zone に含まれている主な文字は次の通りです。

> ラテン文字、発音記号、ギリシア文字、キリル文字、アルメニア文字、ヘブライ文字、アラビア文字、インド諸文字、タイ文字、ラオス文字、グルジア文字、ハングル、各種記号、罫線、仮名

A-zone の最初の 256 文字 (0x0000〜0x00FF) は ISO 8859-1 (Latin-1) と互換になっています。したがって、最初の 128 文字 (0x0000〜0x007F) は ASCII と互換になっています。

A-zone には、結合文字 (combining character) と呼ばれる文字が含まれています。これは、アルファベットのアクセント記号や仮名の濁点のように、他の文字に付加する記号類のことです。たとえば、「a-ウムラウト (ä)」を表すには、「a」のコードの次にウムラウトのコードを置きます。ただし、結合文字を使うと 1 文字のコード長が可変になってしまって扱いにくいので、よく使われる形については合成済み (precomposed) のコードが用意されており、ほとんどの場合は単一のコードで表現できます。ハングルについても、組合せ型コードと完成型コードが用意されています。組合せ型コードでは 1 文字 4 バイトまたは 6 バイトとなります。

I-zone には、中国・日本・韓国 (CJK) の漢字を統合した約 21000 字が含まれています。基となった主な文字集合は、中国の GB 2312、台湾の Big-5、日本の JIS X 0208 と JIS X 0212、韓国の KS C 5601 です。これら各国間では、同じ概念・起源の漢字でも、歴史的・文化的理由により微妙に形が違っている (点画の向きや長さなど) ものが多数ありますが、UCS ではそれらを同一の文字とみなし「統合」しています。ただし、基となった集合で区別されている文字は、互換性のために UCS でも区別されています。

R-zone は特殊用途の領域であり、アラビア文字の変化形や、既存文字集合との互換性を保つための文字などが含まれています。例えば、シフト JIS のテキストには、カタカナや英数字を表現するのに JIS X 0201 (いわゆる半角) と JIS X 0208 (いわゆる全角) の 2 通りのコードがあります。これを UCS に変換する際、半角カタカナと全角カタカナを同じコードに変換してしまうと、それをシフト JIS に逆変換したとき、元のテキストと異なってしまいます。これを避けるため、A-zone の本来のカタカナとは別に、R-zone に「半角カタカナ」が用意してあります。同様に「全角英数字」も用意されています。その他、KS C 5601 などで重複して収録されている漢字についても、重複分が用意されており、可逆な変換が可能となっています。このことからわかるように、UCS は、既存の (単一の) 文字集合/エンコーディング法との互換性に最大限の留意を払っています。

ISO 10646 には、結合文字や組合せ型ハングルの使用を認めるかどうかによって、実装水準という概念が定義されています。

実装水準 1　すべて使用不可
実装水準 2　一部の結合文字 (主にヨーロッパ諸語) と
　　組合せ型ハングルの使用不可
実装水準 3　すべて使用可

Unicode は実装水準 3 です。

また、UCS のすべてを実装するのは困難な場合が多いので、その中から必要な文字だけを取り出したサブセットの使用 (profiling) を認めています。

7.3 UTF

既存のテキストデータ伝送路は、ISO 2022 に準拠した文字データを伝達することを前提にしていますので、そこに UCS のデータをそのまま通すと、制御文字などと誤認されたりして問題が生じます。そのため、UCS に適当なエンコーディングを施して安全に通すことが考えられています。このエンコーディング法はいくつかあり、総称して UTF (UCS Transfer Format) と呼ばれています。これらは、UCS の ASCII 相当部分 (0x0000〜0x007F) は通常の 1 バイトの ASCII に変換し、それ以外のコードをそれぞれの方法でエンコードします。したがって、ASCII テキストの中に UCS (の非 ASCII 文字) を埋め込む方法とみることもできます。

UTF-1

ISO 10646 の付録に収録されている UTF です。UCS-2/4 の 1 文字を、1〜5 バイトの可変長バイト列にエンコードします。エンコード後のバイトデータは、ISO 2022 の GL および GR (0x21〜0x7E、0xA1〜0xFE) の範囲になっているため、ISO 2022 を前提とした伝送路を安全に通過することができます。エンコードには除算を使います。

UTF-1 は、通信や外部記憶などで用いる情報交換用エンコーディング法として設計されましたが、UTF-8 という強力な対抗馬が登場したため、実際に広く使われるようになるかどうかは疑問視されています。

UTF-8 (UTF-2, UTF-FSS)

既存のシステムの多くは、ASCII 図形文字の一部の文字を特殊文字 (区切り文字、エスケープ文字、引用符など) として使っています。このようなシステムには UTF-1 は適しません。UTF-1 は、GL と GR の領域をフルに使っているため、エンコード後のデータの第 2 バイト以降にも GL に相当するコードが現れ、これが ASCII の特殊文字と誤認されて誤動作するおそれがあるからです (シフト JIS の第 2 バイトに 0x5C があるとバックスラッシュと誤認される問題と似ています)。この点を改善したのが UTF-8 です。

UTF-8 は、UTF-1 同様、UCS-2/4 の 1 文字を、1〜6 バイトの可変長バイト列にエンコードします。ただし、UCS の非 ASCII 文字は GR 相当のコード (0xA0〜0xFF) のみ

を使ってエンコードします。これにより、上記のようなシステムでも、ASCII 以外のコードが ASCII と誤認されるおそれはなくなります。また、UTF-8 はエンコード/デコードがビットシフトだけで行えるので、効率の面でも UTF-1 より有利です。

UTF-8 は別名を UTF-2、あるいは UTF-FSS (File System Safe) といいます。FSS という名前は、ファイル名に使ってもパス名の区切り文字 (UNIX ではスラッシュ) と誤認されたりしない、というところからきています。UTF-8 は X/Open やベル研究所によって支持されており、近いうちに 10646 に収録されて UTF-1 にとってかわるだろうというのが大方の予想です。

UTF-7

UTF-1 や UTF-8 は 8 ビットの伝送路を前提としていますが、インターネットではいまだに 7 ビットしか通さない通信路も珍しくありません。また、メールのヘッダ部などでは、ASCII 図形文字の使用すら制限されていることがあります。そのような状況でも UCS をエンコードするために考案されたのが UTF-7 です。

UTF-7 は、MIME の Base64 エンコーディングや uuencode と類似の方法によって、UCS の非 ASCII 文字を ASCII 図形文字にエンコードします。エンコードされた非 ASCII 部分の前には + を、後ろには - を置くことによって、「地」の ASCII テキストと区別します。

UTF-7 はもともと上記のような制限の強い状況で使うためのものであり、広範囲に使われることを目的とした UTF-1 や UTF-8 とは性格が異なります。

UTF-16

これは、UTF と名付けられてはいますが、目的も内容も他の UTF とは全く異なっています。他の UTF は ASCII テキストに UCS を混ぜるためのものですが、UTF-16 は UCS-2 のテキストに UCS-4 (の一部) を埋め込むためのものです。したがって、他の UTF と併用することもできます。

UTF-16 では、UCS-4 のうち BMP の次の 16 面 (総文字数 1 メガ) のコードを、O-zone のコード 2 個に変換して UCS-2 のテキストに埋め込みます。これにより、UCS-2 で表現可能な文字数を大幅に増やすことができます。

しかし一方で、1 文字 2 バイト固定の原則の例外を増やすことになりますし、何より UCS-4 が BMP 以外まだ何も決まっていない現状ではやや先走りの感は免れません。BMP (UCS-2) はコード空間が狭い、という批判をかわすための苦肉の策という気もします。

Linux お気楽・極楽デバイスドライバ

Sumii

1 はじめに

いまさら説明するまでもないと思いますが、Linux というのは Linus Torvalds さんが中心になって作られたフリーな PC UNIX です。今日はその Linux 用のデバイスドライバを作る方法の概略を紹介したいと思います。

DOS ならともかく UNIX 系 OS のデバイスドライバを書くとなると、何だか難しいように思えるかもしれませんが、少なくとも Linux に関してはいたって簡単ですので、心配する必要はありません。Linux と C 言語に関する基本的な知識があれば十分理解できる内容です。

なお、この文書が対象とするカーネルのバージョンは 2.0.0 以上です。それ未満のバージョンでも基本的なところは同じですが、細かい点が微妙に異なるので、そのままではうまくいかない可能性があります。

2 とりあえずやってみる

いろいろ説明する前に、まずは実際に例を試してみることにしましょう。次のソースを入力して「counter.c」というファイル名でセーブしてください。短いプログラムですから、たとえ手で入力してもすぐに終わりますよね。

```
 1  #define __KERNEL__
 2  #define MODULE
 3
 4  #include <linux/errno.h>
 5  #include <linux/fs.h>
 6  #include <linux/module.h>
 7  #include <linux/mm.h>
 8
 9  static unsigned int counter = 0, busy = 0, eof = 0;
10
11  static int counter_open (struct inode * inode, struct file * file) {
12    if (busy) return - EBUSY;
```

[0] 理学部情報科学科三年　住井 英二郎 (sumii@is.s.u-tokyo.ac.jp)

```
13
14    busy = 1;
15    MOD_INC_USE_COUNT;
16
17    eof = 0;
18
19    return 0;
20  }
21
22  static void counter_release (struct inode * inode,
                                 struct file * file) {
23    busy = 0;
24    MOD_DEC_USE_COUNT;
25  }
26
27  static int counter_read (struct inode * inode, struct file * file,
28                           char * buf, int count) {
29    char str [16];
30    int len, i;
31
32    if (eof) return 0;
33
34    sprintf (str, "%d\n", counter ++);
35    len = strlen (str) + 1;
36
37    i = verify_area (VERIFY_WRITE, buf, len);
38    if (i) return i;
39
40    memcpy_tofs (buf, str, len);
41
42    eof = 1;
43
44    return len;
45  }
46
47  static struct file_operations counter_fops = {
48    NULL, /* counter_lseek */
49    counter_read,
50    NULL, /* counter_write */
51    NULL, /* counter_readdir */
52    NULL, /* counter_select */
53    NULL, /* counter_ioctl */
54    NULL, /* counter_mmap */
55    counter_open,
56    counter_release,
57    NULL, /* counter_fsync */
58    NULL, /* counter_fasync */
59    NULL, /* counter_check_media_change */
60    NULL, /* counter_revalidate */
61  };
62
63  int init_module (void) {
64    int i;
65
66    i = register_chrdev (63, "counter", & counter_fops);
67    if (i) return - EIO;
```

```
68
69    return 0;
70  }
71
72  void cleanup_module (void) {
73    unregister_chrdev (63, "counter");
74  }
```

入力できたら、以下の操作を行ってコンパイルと組み込みをします。root でないと実行できないコマンドも含まれていますが、この文章をお読みのみなさんはもちろん root でしょう。:-)

```
% gcc -Wall -O2 -c counter.c
% su
Password:
# insmod counter.o
# mknod -m 444 /dev/counter u 63 0
# exit
exit
```

うまくいったら「cat /dev/counter」を繰り返し実行してみてください。どうですか、おもしろいでしょう？(つまらなかったらごめんなさい。)

「どこか『デバイス』ドライバなんだ」という話もありますが、基本的な枠組みは本当のデバイスドライバでも同じですので、以下はこれを例として説明していきたいと思います。このように必ずしも実際のハードウエアを伴わないデバイスを仮想デバイスといいます（やや嘘)。

3 今やったのは何？

さて、ほとんど何の説明もせずに、いきなり実際にドライバをコンパイルして組み込んでみました。でも、これではわけがわかりません。先ほどの操作は、いったいどういう意味だったのでしょうか。順を追って説明していきます。

```
% gcc -Wall -O2 -c counter.c
```

counter.c をコンパイルして、counter.o を作ります。

ポイントは、オブジェクトファイルのみを作って、実行ファイルは作らないという点です。実はこのドライバは「installable kernel module」という形式をとっています。「installable kernel module」というのは、カーネルを再構築したり、システムを再起動したりせずに、カーネルに新しい機能を追加するものです。ダイナミックリンクライブラリと似ていますが、ユーザプログラムが使用するライブラリではなく、カーネルに組み込まれるモジュールであるという点が異なります。あくまでモジュールなので、実行ファイルではなくオブジェクトファイルなのです。

```
# insmod counter.o
```

たったいま作成したモジュールを組み込みます。現在どういうモジュールが組み込まれているかは「lsmod」というコマンドでわかります。モジュールを削除するには「rmmod」というコマンドを使います。

```
# mknod -m 444 /dev/counter u 63 0
```

counter のためのデバイスファイルを作成します。メジャー番号 63 番、マイナー番号 0 番のキャラクタデバイスです。

メジャー番号は割り当てが決まっているので、本来勝手に使ってはいけないものです。割り当ては、カーネルソースに付属している Documentation/devices.txt に書かれています。ただし、それをよく読むと

```
60-63  LOCAL/EXPERIMENTAL USE
       Allocated for local/experimental use.  For devices not
       assigned official numbers, these ranges should be
       used, in order to avoid conflicting with future assignments.
```

とあるので、60〜63 番はこのような実験に使っても良いのです。

4 ソースの説明

```
 1  #define __KERNEL__
 2  #define MODULE
 3
 4  #include <linux/errno.h>
 5  #include <linux/fs.h>
 6  #include <linux/module.h>
 7  #include <linux/mm.h>
```

必要なヘッダファイルをインクルードします。カーネルに組み込むモジュールであることを表すために、__KERNEL__ および MODULE を #define します。この#define はヘッダファイルの中で参照されているので、#include より前でしなければなりません。

```
 9  static unsigned int counter = 0, busy = 0, eof = 0;
```

モジュール内部で使用する変数を宣言します。

先頭の「static」は重要です。組み込み時の名前衝突を回避するために、モジュールの外部から参照されない名前は、すべて static にしなければいけません。

各変数の意味は後になればわかります。

```
11  static int counter_open (struct inode * inode, struct file * file) {
12    if (busy) return - EBUSY;
13
14    busy = 1;
15    MOD_INC_USE_COUNT;
16
17    eof = 0;
```

```
18
19    return 0;
20  }
```

デバイスファイル（/dev/counter）が open されると呼ばれる関数を用意しておきます。何やら二つほど引数がありますが、今回は関係ないので気にしません。(^^;

一度に多数 open されるとややこしいので、busy という変数を使って排他制御を行ないます。「これじゃあ排他制御になってないじゃん」と思うかもしれませんが、Linux ではカーネルモードのプロセスはプリエンプトされないので問題ありません（と思う）。

MOD_INC_USE_COUNT は、モジュールが使用中であることを示す値を一つ増やします。この値が 0 でなければモジュールを削除できないので、誤って使用中のモジュールを削除してしまうおそれがなくなります。

```
22  static void counter_release (struct inode * inode,
                                 struct file * file) {
23    busy = 0;
24    MOD_DEC_USE_COUNT;
25  }
```

デバイスファイルが close されると呼ばれる関数を用意しています。
MOD_DEC_USE_COUNT は、MOD_INC_USE_COUNT の逆です。これを忘れると、モジュールがずっと使用中になって、削除できなくなってしまいます。

```
27  static int counter_read (struct inode * inode, struct file * file,
28                           char * buf, int count) {
29    char str [16];
30    int len, i;
31
32    if (eof) return 0;
33
34    sprintf (str, "%d\n", counter ++);
35    len = strlen (str) + 1;
36
37    i = verify_area (VERIFY_WRITE, buf, len);
38    if (i) return i;
39
40    memcpy_tofs (buf, str, len);
41
42    eof = 1;
43
44    return len;
45  }
```

デバイスファイルが read されたとき呼ばれる関数を用意します。このドライバの中核部です（笑）。現在の counter の値を 10 進数の文字列に直し、それをユーザプロセスが用意したバッファにコピーします。

ここで注意しなければならないのは、ユーザプロセスが用意したバッファはユーザメモリ空間にあるので、普通のポインタを使ったメモリ操作ではなく、専用のマクロ（memcpy_tofs）を使って書き込みを行うということです。また、不当な領域に書き込みを行わないよう、あらかじめ verify_area という関数で、渡されたアドレスの正当性をチェックしなければいけません。

なお、ここで使っている sprintf と strlen は（名前と意味は同じでも）決して普通のライブラリ関数ではなく、モジュールのためにカーネルがエクスポートしている関数、ないしはヘッダに含まれているマクロです。

```
47  static struct file_operations counter_fops = {
48    NULL, /* counter_lseek */
49    counter_read,
50    NULL, /* counter_write */
51    NULL, /* counter_readdir */
52    NULL, /* counter_select */
53    NULL, /* counter_ioctl */
54    NULL, /* counter_mmap */
55    counter_open,
56    counter_release,
57    NULL, /* counter_fsync */
58    NULL, /* counter_fasync */
59    NULL, /* counter_check_media_change */
60    NULL, /* counter_revalidate */
61  };
```

デバイスファイルを操作したとき呼び出される関数を登録するための構造体です。次で説明するデバイスの登録のとき使用します。lseek やらmmap やらいろいろなインターフェースが用意されていますが、使わないところは NULL で構いません。

```
63  int init_module (void) {
64    int i;
65
66    i = register_chrdev (63, "counter", & counter_fops);
67    if (i) return - EIO;
68
69    return 0;
70  }
```

モジュールを組み込んだときに呼び出される関数です。したがって、この関数は static にしてはいけません。また、名前もこの通りにしなければなりません。

register_chrdev で、メジャー番号 63 番の counter という名前のデバイスを登録します。ちなみに、現在登録されているデバイスの一覧は「cat /proc/devices」で見られます。

```
72  void cleanup_module (void) {
73    unregister_chrdev (63, "counter");
74  }
```

モジュールを削除したときに呼び出される関数です。init_module と同様に、static にしてはならず、名前も変えてはいけません。

init_module で登録したデバイスを、unregister_chrdev で忘れずに削除しておきます。

5 モジュールのための関数とマクロ

counter の説明は以上です。簡単ですね。でも、実際に何かのボードのデバイスドライバを書こうとしたら、さすがにこれだけではすみません。I/O アクセスを行なったり、物理メモリや IRQ を確保したりするにはどうすればいいのでしょうか？

実は先ほどの `sprintf` や `strlen` のように、カーネルがモジュールのために提供している関数や、ヘッダに含まれているマクロがたくさんあります。カーネルが提供する関数については「`ksyms -a`」で一覧が見られます。I/O アクセスやリソース確保を行うときには、これらの関数・マクロを使います。

残念ながら、これらの関数・マクロの説明はあまりありません。一部の重要な関数・マクロについては、manpages の 9 章 (!) に書かれていますが、載っている数は決して多くない上に、情報も古いのであまりあてになりません。ヘッダやカーネル、ドライバのソースを見て理解するしかないでしょう。

もちろんネットワーク上にも情報があります。あまりちゃんと探したわけではありませんが、筆者が知っている範囲では

`http://speedy.redhat.com:8080/HyperNews/get/devices/devices.html`

が参考になるでしょう。

6 おわりに

何だかいろいろと偉そうなことを書きましたが、筆者自身つい最近知ったことが多いので、ひょっとしたら間違っているところもあるかもしれません。もし誤りがあったらメールや italk で教えていただけると幸いです。:-)

初心者のための 8086 講座

第 3 回

高野商店

どうも高野商店です。前回はなんか薄い内容だったので、それじゃあつまらんと言うことで突然ですが gas やります。

1 gas とはなんぞや？

gas とは GNU assembler のことで、その名の通り GNU が作ったアセンブラーです。gas は gcc (GNU C compiler) が吐いたアセンブラーコードを処理するアセンブラーであり、gcc をインストールすると漏れなくついてきます。:-) gas にはいろいろなプロセッサ用がありますが、ここでは 386 用の gas について述べます。gas は MASM (Microsoft macro assembler) などに比べて次のような利点があります。

1. プロテクトモード対応である。
2. それ故、メモリの制限が無い。(つまり 1MB の壁を考える必要はない)
3. もちろん、386、486、の独自命令をサポート。

まあ 1. が言えれば 2. 3. は当たり前なのですが... (^^;;
欠点としては

1. マクロが使えない。
2. リアルモード用の 16 ビットアドレスをサポートしていない。

ぐらいです。1. は gas があくまでもアセンブラーに徹しているという性格上仕方のないものです。しかし、gas は gcc と抱き合わせてインラインアセンブラーとして使えるのであまりマクロの必要性は感じられません。

2 gas の記述方

1. ニーモニックはほぼ intel 社の表記法と同じで、オペランドがバイト/ワード/ロングなどを示す b/w/l を後に付けます。
2. 前回ではニーモニックのあとにつけたオペランドの順番はディスティネーション、ソースの順番でしたが、gas の場合これが逆になります。
3. レジスタの名前には頭に
4. immediate オペランド (つまり直接値) には頭に $ をつける。
5. 絶対番地オペランドには頭に * をつける。
6. メモリ参照は

   $$segment\_register : displaysment(base\_register, index\_register, scale)$$

   となります。

さて、以上のようなことをふまえて前回書いたアセンブラーを gasっぽく書き換えてみましょう。

```
MOV AL,16H      -> movb $0x16, %al  //0x は 16 進法を表しています。
MOV [0541H],AX  -> movw %ax, *0x0541
ADD AL,2AH      -> addb $0x2a, %al
MOV AX,[BX+DI+4]-> movw %cs:0x04(%bx,%di,2), %ax
```

となります。

このようにほとんど intel 社の命令群と大差がないのですが、info を読むとごく一部の命令に違いがあることがわかります。その差は後々になってからお話しします。

3 gas

実際に gcc に gas のコードを吐かせてみましょう。

```
.text
LC0:
        .ascii "Hello World!!\12\0"
        .align 4
.globl _main
_main:
```

```
        pushl %ebp
        movl %esp,%ebp
        call ___main
        pushl $LC0
        call _printf
        addl $4,%esp
        xorl %eax,%eax
        jmp L1
        .align 4,0x90
L1:
        leave
        ret
```

これは C 言語でお馴染みの Hello.c です。(注　ここで使っているのは 8086 のアセンブラなので情報棟の gcc でやっても動きません。305 の Xa でやってみて下さい。) ちなみに .align とかはコンストラクタと呼ばれ、i386/387 の命令を発生する以外の直接 gas に対して命令を送るものです。また、レジスタの頭に e のついているもの (%eax など) は 32 ビットレジスタを意味します。

4　解説

まず代表的なコンストラクタを説明します。

1) .text　命令部に入れることを定義します。
2) .globl　ラベルを他のモジュールから参照可能にする。
3) .align　.align n で次の文を 2^n の倍数の番地に置く。
4) .ascii　label: .ascii "**文字列**" の形式で文字列を定義する。

上でとりあえず使っているのは以上の 3 つだけです。ところで 3) はどんな意味があるのでしょうか？ 上の例では .align 4 となっていますが、gcc によって出力されたアセンブラーのリストはすべての関数の頭にこれがついてます。これは 4 バイトごとにメモリにデータが割り当ててあると 4 バイトのデータ (long 型のデータ) や 8 バイトのデータ (double 型のデータ) を高速にアクセスできるようになるわけです。その分メモリの空きが多くなるのでメモリをバカ食いします。(^^;

さてここで初めて出てきた call と jmp について説明します。

5 ジャンプ文

プログラムは常にメモリアドレスが低いところから高い方に向かって格納されていますが、途中でデータの結果によって処理の仕方を変えなければならないことが起きてきます。ジャンプ命令には無条件ジャンプと条件ジャンプとがあります。ジャンプ命令を実行すると、いままで通り次のメモリに入っている命令を実行するのではなくジャンプ命令で指定したアドレスへ飛んでいきます。無条件ジャンプはどんな場合でも指定されたアドレスにジャンプします。条件ジャンプはある条件が満たされたときのみジャンプします。条件が合わなかった場合、普通に次の命令を実行します。ところでその条件とはいったいなんでしょうか？ 以前、説明したフラグレジスタを条件材料として条件ジャンプを実行するわけです。これによって C 言語でよく用いられる if 文等の条件分岐が可能になります。

まずは無条件ジャンプです。gas では

`jmp` *label*

のように書きます。*label* は上の例では `L1:` のように「ラベル名:」で指定します。これを実行すると、無条件に `L1:` 以下の命令を実行します。`cmp 0x02,%al` を実行したあと

命令		結果
`jz` *label*	(Jump if Zero)	`al==0x02` ならばジャンプ
`jnz` *label*	(Jump if Not Zero)	`al!=0x02` ならばジャンプ
`jb` *label*	(Jump if Below)	`al<0x02` ならばジャンプ
`ja` *label*	(Jump if Above)	`al>0x02` ならばジャンプ
`jbe` *label*	(Jump if Below or Equal)	`al<=0x02` ならばジャンプ
`jae` *label*	(Jump if Above or Equal)	`al>=0x02` ならばジャンプ

となります。上に挙げた例は 2 つのを値をそれぞれ符号なしの 1 バイトデータとしてみなした場合です。符号ありの場合は最上位ビットある無しで負の数かそうでないかを決定します。だからたとえば 0xff は符号無しの場合は 255 を表し、符号ありの場合は -1 を表します。下の命令は符号ありの場合の条件ジャンプです。

命令		結果
`jl` *label*	(Jump if Less)	`al<0x02` ならばジャンプ
`jg` *label*	(Jump if Greater	`al>0x02` ならばジャンプ
`jle` *label*	(Jump if Less or Equal)	`al<=0x02` ならばジャンプ
`jge` *label*	(Jump if Greater or Equal)	`al>=0x02` ならばジャンプ

符号あり無しの話をもう少し詳しくしてみます。符号無しの場合、16 進法と 10 進法との対応関係を考えてみると

16 進法: 00,01,02, ...,0d,0e,0f,10, ..., fd, ff
10 進法: 0, 1, 2, ...,13,14,15,16, ...,254,255

これが符号ありの場合になると...

16 進法: 80, 81, 82, ...,fe,ff,00,01, ... ,0e,0f,10, ..., 7e, 7f
10 進法: -128,-127,-126, ...,-2,-1, 0, 1, ... ,14,15,16, ...,126,127

まったく違うことがわかります。だからもし

```
mov 0xff,%al
cmp 0x02,%al
```

としたあと、jb と jl は全く違った意味を持つことになるわけです。

6 コール文

さて次はコール文です。プログラムは普通、低いところから高いところのアドレスに向かって実行されていきますが、その中で同じことを何度も書かなくてはいけなくなるなるときがあります。C 言語ではこれを一つの関数としてまとめ、必要なところでその関数を呼び出します。このようにプログラム本体とは別の場所に作られ、プログラム本体から呼び出されるためのあるまとまった処理を行うプログラムのことをサブルーチンと呼びます。プログラム本体のことをメインルーチンと言います。上の例では __main と _printf がサブルーチンとして呼び出されています。call _printf と呼び出されている部分がそれです。ここには _printf サブルーチンは書いてありませんが、それはどこにあるのでしょうか？ _printf や ___main サブルーチンはライブラリーの中に納められていて、C 言語ではリンクする段階で一緒に結合されます。だから、アセンブラーの中に書いて無くても大丈夫な訳です。

サブルーチンが必要なもう一つの理由としてはプログラムを見やすくするためです。たとえ一度しか実行しない処理であっても、ある程度まとまったものならサブルーチンにしてしまった方が、プログラムの構造がわかりやすくなり、デバッグ等の作業も楽になります。

CPU の命令では、メインルーチンからサブルーチンを呼び出す call (コール) 命令と、サブルーチンからメインルーチンへ戻る ret (リターン) 命令が用意されています。この二つは常に対になって用いられます。簡単な例を出してみると...

```
#include <stdio.h>
int func(int n)
{
        return n*10;
}

void main()
{
        int n;
        n = func(2);
        printf("%d",n);
}
```

むう、なんかめちゃくちゃつまらないプログラムですが (^^; これをアセンブラに翻訳すると...

```
.text
        .align 2
.globl _func
_func:                          /*func(int) の本体*/
        pushl %ebp
        movl %esp,%ebp
        movl 8(%ebp),%eax       /*ここで引数のロードをしている*/
        movl %eax,%edx
        sall $3,%edx
        addl %eax,%edx
        leal (%eax,%edx),%ecx
        movl %ecx,%eax
        leave
        ret                     /*メインルーチンに戻る*/
LC0:
        .ascii "%d\0"
        .align 2
.globl _main
_main:                          /*メインルーチン。ここから実行される*/
        pushl %ebp
        movl %esp,%ebp
        subl $4,%esp
        call ___main
        pushl $2
        call _func              /*ここで func(int) を呼んでいる*/
        addl $4,%esp
        movl %eax,%eax
        movl %eax,-4(%ebp)
        movl -4(%ebp),%eax
        pushl %eax
        pushl $LC0
        call _printf
        addl $8,%esp
        xorl %eax,%eax
        leave
        ret
```

そのまま gcc に吐かせたコードではなく少しいじくってあります。

上で見てもらってもわかるように call で呼び出されて、ret で再びもとに戻ってき

ていることがよくわかります。もちろん _printf なども実際にはライブラリに ret 命令が記述してあるわけです。

さて、関数 func には引数として 2 が渡されているわけですが、この引数はアセンブラー上ではどのように実現されているのかを見てみます。うえのメインルーチンで怪しそうなところを見ていくと pushl $2 という命令が call _func の前にあるのがわかります。push 命令は、前回スタックの操作のところでやりましたが、gcc では引数を渡すのにこのスタックの操作をすることによって実現しているわけです。サブルーチン _func を見てみると pop 命令を使ってではなく、sp (スタックポインタ) を参照して mov 命令を使って引数を参照する手段を取っています。ここで bp (ベースポインタ) を使ってメモリを参照していますが、セグメントレジスタを指定していません。一般に bp を指定した場合、セグメントレジスタは自動的に ss (スタックセグメント) になります。

7 スタックについての復習

スタックは元来アドレスを気にせずに簡単にデータの出し入れができるようになっている仕組みのことです。スタックにデータを積むと、最初に取り出せるのは最後に積んだデータです。スタックにデータを積む命令が push 命令で、スタックからデータを取り出す命令を pop 命令と言いました。sp はこのスタックの一番上に積まれているデータのアドレス (オフセット) を指すようになっています。たとえば ax レジスタをスタックに出し入れする命令は

```
pushw %ax
popw %ax
```

です。またフラグレジスタに関しては特別に

```
pushf
popf
```

というフラグ命令が用意されています。

さて、次回は実際に gas をインラインアセンブラとして使う方法と、アセンブラ命令の続きをやります。

コミケ

高野商店

みなさんはコミックマーケット (通称コミケ) というイベントをご存じでしょうか？ マンガやアニメなどが好きな人ならば、名前ぐらいは聞いたことがあるかもしれません。コミックマーケットとは「マンガ、アニメ、ゲーム、小説及びその周辺ジャンルにおける表現の可能性を追求する場としての同人誌。営偽の結実としての作品。それを一般に向けてアピールする場」なのです。つまり、簡単に言うと「自費出版の同人誌を売り買いしたり、交流を深める場所」なのです。

こう書くとなんかパソコンとは全く関係ないように聞こえますが、実はここではマンガやアニメの同人誌以外にも、パソコンの同人誌や自作ハード、同人ソフト等の販売も行われてます。

また同人誌は自費出版という性質上、記事の内容に関して広告企業からとやかく言われることは無いので、かなりやばい裏情報とかを載せた本が売っていたりします。(^^; このコミックマーケットは年に夏と冬に 2 回行われ、今年は 8/4, 5 に夏のコミックマーケットが開催されました。場所は今回から有明のビッグサイトで開かれることになりました。[1)]

私がコミックマーケットに来るのは今回で 3 回目だったのですが、会場が新しくなったせいで去年と比べると大分快適でした。おそらく冷房の利きが良かったせいかもしれません。[2)] ただ例年道理 2 日間で、のべ約 30 万人の人が全国から集まってくるので、相変わらずの混みようでした。コミケ初心者の方はすきはじめる午後から来るのが得策でしょう。(ただ午後から来ると結構完売してしまっている本とかソフトとかあるんですけどね。(^^;)

さて、実際どんな本が売られていたのでしょうか？ 今回私が買ってきた中でこれはと思えるような本をピックアップしてみました。

「さらば 98 Ver.1.2」　オフセット本

去年の冬のコミックマーケットに出されていた本のバージョンアップ版。初代 PC-9801 から最新の Can Be にいたるまで、すべての 98 を一台一台取り上げ、そのマシンの解説、批評が書いてあります。詳しいスペックとかは載っていないのですが、ツボを押さえており、読んでいてなかなか愉快です。一般の 98 以外にも、かなりマニアックなマシンの紹介や (FC-98, SV-H98 等)、98 の悪口等も載っています。

例えば NEC 最大の悪行と言われている 55 ボードの話。「intel inside」の代わりに一

[1)] 前回までは晴海の国際展示場。
[2)] 晴海は冷房の利きが悪かった。

時期「intel in it」が代わりに使われていた秘密。など、コラムとしてまとめてあります。
巻末には 98 の系譜も載っています。
表紙を逆転すると「あばよ 286, 386, 486, 586」と印刷されており、こちらは EPSON の 98 互換機マシンを取り上げています。一読の価値がある本です。

「趣味の秋葉本」　コピー本

パソコン雑誌ではパソコンショップの地図しか載せてくれないことが普通です。この本はパソコンショップ以外の知られざる秋葉原の名 (迷？) 所を紹介した本です。パソコンジャンク編とカレーショップ編、飲料自販機編があります。カレーショップ編では各カレーショップの値段が載っていたりしてなかなか実用的です。また、秋葉原では怪しい飲料自販機が結構あるので、その一例としてポッカ倶楽部の自販機の紹介や、チェリオの自販機の紹介などが写真入りで載っています。
皆さんは５回以上振ってから飲むプリンシェイクとか、おでんとかの缶ジュースが存在することを知っていましたか？[3)]
秋葉原での買い物の際の心強いナビゲーターとなってくれるでしょう。

「X-1 平和研究所」　オフセット本

その名の通り今はなき X-1 の活用を中心に書かれた本。ハードの制作から、初心者向けの記事までしっかり書かれています。中古パソコン (特に 8 ビット機) について熱く書かれています。[4)] ここでは私は記念撮影をしてもらいました。(^^;;;

「ゲームアホ」　コピー本

「ゲームアホ」とはもちろん「ゲームラボ」のパロディーです。本家の「ゲームラボ」誌も相当怪しいネタや改造が載っていたりするのですが、この本もそれに負けないぐらい怪しいです。(笑)
普通ファミコンはファミコンのケースの中に入っているのはみなさんご存じの通りですが、これをあの小さなモデムのケースの中に入れてしまおうという、とんでもない記事が載っていました。実際に完成品がブースの所に置いてありました。カセットのカートリッジは後ろから差し込むようになっていて、ちゃんとジョイスティックの端子もついています。(笑)
それ以外にもコントロールボックスを自作して基板でゲームをしようという実用的な (?) 記事も載っています。
「スーパーファミコンでも同じことをやらないのか？」との問いに「そこまでお金がない。」とのことでした。(^^;;

[3)] 本当にある。
[4)] (ちなみに今回のコミケで X-1 用のソフトを出していたのはここだけだった。)

「裏ディクショナリー」　オフセット本

前半は大手ゲームメーカー、セガの裏事情を中心に座談会の形で書かれています。伏せ字が多いのですが、ゲーム事情に通じている人ならばこのくらいは解読できてしまうのでは？ 特に上司に対する恨み辛みがすごくて、つい「ここまで書いて大丈夫かよ。」とつっこみを入れたくなるほどやばいことが書いてあったりします。

後半はかなりブラックユーモアの効いた悪口が大量に載せてあります。

黒い噂や、黒い冗談が大好きな人にお勧めの一冊。(笑)

「MSX ALHAMBRA No.1,2」　オフセット本　　FD 付録

本の名前は MSX ですが、記事の内容は MSX に限定されず、CG やハードの制作、プログラミング等が載っています。ページ数の割には割高 (No.1 1100 円 No.2 1000 円) のように見えますが、FD 付録なのでこの価格は納得。

この本を出してるサークルは将来的に株式会社を作り、自分たちの手で MSX を製造する計画があるそうです。また、MSX オンリーの同人ソフト即売会等の MSX のための啓蒙活動を行っています。MSX ユーザーの人々には心強いサークルですね。

以上のような怪しくも優れた同人誌ほかにも同人ソフトの販売や、ハードの販売も行われていました。

特に今回目を引いたのは、参考展示として X68030 を改造した X68060 の展示でした。つまり CPU を 68030 から 68060 に改造してあるのです。68040 への改造は本で出版されているのでそれほど驚かないのですが、やはり X68060 というと風格が違います。聞いた話ではすでに 1 年半ぐらい正常に動いているとのことでした。

他に珍しいパソコンとして、アップルが作った限定品のパワーブックがデモに使われていました。これはアップルが初めて主催した女子ゴルフが開催れた時に作られた限定品らしいのですが、液晶はモノクロであるにも関わらず、ケース本体がすごいカラフルなパワーブックです。売り子さんの話では、こういうイベントの時には非常に目立って、役に立っているそうです。

全く役にはたたないものなのですが、マニア心をくすぐる品物も売っています。DRAM でできたネックレスがそれです。パリありとパリ無しの 2 つあったのですが、家の Xa はパリありなので前者の方を買いました。[5] これをつけてこの後買い物をしていたら、なかなか売り子さんにうけが良かったです。

DTM で作曲した CD を売っているところや、CG 集、怪しいゲーム等 (笑) も売られていました。中には開場 1 時間後には売れ切れているゲームソフトもありました。

MSX や X68000 シリーズはすでに製造されていないため、雑誌等で情報を得るのは難

[5] 1000 円の所を値切って 900 円に負けてもらいました。^^;)

しくなってきています。[6] そのためユーザー間のつながりが強く、今回のコミケでも Mac や Windows に押されてきているとはいえ、まだまだ多くのハードやソフトの販売がされていました。

今回たくさんの売り子の人に話しかけてみたのですが、タダでいらなくなったジャンク品を貰えたり[7]、別の場所では甘食も貰えたりします。(^^;;; おかげで楽しい一日を過ごすことができました。

コミケは単なる即売会と言うよりも、むしろお祭り的要素を持った交流会と言った方が正しいのかもしれません。

次回、冬コミは 12/28,29 にあるのですが、興味のある方は行かれてみてはどうでしょうか？ もしかしたらお気に入りのソフトや本、CG 集などが見つかるかもしれませんよ。

補足

このレポートは 2 日目の一部です。1 日目はうえ君と回ったのですが、うえ君もすっかり毒されてしまったようです。(^^;;; わたしは前日に新幹線をつかって 12 時 30 分には上野に帰っていたのですが、夜の 10 時まである労働にかり出されたのでした。(T_T) 2 日目は別の友人と行ったのですが、1 日目より混んでいました。それでも去年の地獄のような暑さ+3 日間に比べれば天国でしょう。おかげで今年はあんまり体重減らなかったし。(爆) ただ外で並んでいる時に突然列が移動してしまったのは何ともまぬけでした。文句を垂れているねーちゃんどもが非常に愉快でした。:-) kms さんが徹夜明けで非常に恐かったです。(笑)

[6] 先日 Oh!X も廃刊になってしまいましたし...
[7] わたしはインテルの 385 をタダで頂いてきました。

頭の体操

すべての問題には証明が必要です。解答はそのうち掲載。解けた人は編集長まで。

Q. 1 (数セミより)

短針と長針しかない時計を考えると、針を見ても時刻はわからない。例えば、短針と長針が重なっている時計をみても、12 時、1 時 $5\frac{5}{11}$ 分、2 時 $10\frac{10}{11}$ 分などの可能性があるので、目盛なしでは時刻の特定はできない。

では、短針と長針と秒針のある目盛なしの時計を与えられたとき、時刻の特定は可能か否か。

Q. 2 (油すまし)

2 次元ユークリッド空間における、中心 $(0,0)$ 半径 2^{1996} の円周上に、格子点 (座標が整数で表される点) はいくつ存在するか。

Q. 3 (油すまし)

10 個の赤玉と 10 個の白玉がある。これをよく混ぜ、10 個ずつ 2 つの袋に分ける。2 人が順番に好きなほうの袋から玉を 1 つ取り出す。赤玉を当たりとするとき、2 番目の人の戦略として、

(A) 1 人目の人が赤玉を引いたら同じ袋から、白玉を引いたらもう一方の袋から取る。

(B) 1 人目の人が白玉を引いたら同じ袋から、赤玉を引いたらもう一方の袋から取る。

どちらがどれだけ有利か。

Q. 4 (油すまし)

2 つのサイコロの各面に 1 つずつ 1 から 12 までの数字を書く。このサイコロ 2 つを振ったときの大きいほうの目の期待値を最も大きくするためには、どのように書きこめばよいか。

Q. 5 (油すまし)

A，B，C の 3 人がじゃんけんをして勝者 1 人が賞品をもらえることになった。そこで、A，B の 2 人はあらかじめ示し合わせて有利になるように戦略を立てることにした。

A，B が最も有利になるように戦略を立てたとき、A，B いずれかが賞品を得る確率はどれだけか。

編集後記

ついに部報も 200 号に達しました (^_^)。

今回から編集環境を Linux に移行。出力も dviout から dvi2ps + ghostscript に移行しました。PC UNIX 環境はなかなか愉快ですね。

と調子に乗って画像をどんどん EPSF にして張り込んだら ps ファイルが 5MB を突破してしまった (^^;;

表紙のロゴの PostScript ファイルは Tellur さんに作っていただきました。

次は 201 号「秋休み号」の予定です。

理論科学グループ　部報 200 号

1996 年 8 月 23 日　発行

発行者　金子　済

編集者　大岩　寛

発行所　理論科学グループ

〒 153　東京都目黒区駒場 3–8–1

東京大学教養学部内学生会館 305

Telephone: 03–5454–4343

理論科学グループ部報 第 200 **号**

— 夏休み号 —

1996 **年** 8 **月** 23 **日**

THEORETICAL SCIENCE GROUP