Theoretical Science Group

理論科学グループ

部報 196 号
— クリスマスコンパ号 —

目　次

TSG 代替わり特集

TSG役員人事

役職	'95 年度	'96 年度
部長	NAO	わたる
副部長	しげ	高野
会計	nishi	村井
編集長	ちょもらんま	おおいわ
副編集長	k_o_	げる
ライブラリアン	gana	げる
コンパ委員	sigma	かもしま
	Yu	ZERO
	しげ	きりもみ
		窓明
学館連絡員	nishi	高野
ICCC 連絡委員	hanawa	
庶務	Yu	はと
		竹島

TSG旧役員離任挨拶

'95 部長　NAO

前部長のなお、こと渡辺尚貴です。1年間の部長の役職を何事もなく無事まっとうできました。今は幸せな気分で大御所生活を送っています。

大御所様なので、優雅な暮らしなのかと思われるかも知れませんが、実は物理学科の授業や演習やレポートやテストで日が暮れる一階の普通の学生にもどってしまったのです。

部長の権威を使って授業を全部切っていたころが懐かしいです。

昨年はコンテストに一切出場しなかったかわりに、いろいろな雑誌社の記者が取材に305にやってきました。しかし、彼らの対処はすべて同じ編集者な編集長の安田君がしてくれました。

というわけで、昨年の部長の仕事はオリと駒場祭に限られました。サークルの長たる部長の職務として一般に言われているものは、サークルの運営の統括・指揮なのですけれど普段はそんなことをする必要はありませんでした。サークルTSGはコンピュータ好きな連中がわいわい集うところなのです。なにも部長が細部の動きまで指令する必要は全然ありません。みんな各自が好きなように遊んでいられれば良いのです。

部長の仕事は、みんながこんなふうに好きなように遊んでいられる環境を保持することなのです。具体的にはクラス・サークル代表者会議（CC代）を主とする数多くの会議に出席して書類を整理することです。書類の中には、間違いなく提出しないとTSGが滅んでしまうようなものもあり、なかなか神経を使いました。

オリや駒場祭の前後では、いろいろな準備の指揮をとります。人材の確保、準備の進行状況、機材の移動、部屋の環境このときは急にいそがしくなります。

でも基本的に部長の仕事は確認と指令のみです。いそがしくても体をつかった忙しがしさではないので、この点はコンパ委員や編集長よりもずっと楽な仕事です。

昨年度の部長は、TSGの部長であるのに、コンピュータに関する知識はかなり低いものでした。それでも部長を務めることはできました。それはなぜでしょうか？

「理論科学グループ」、太古の昔は本当に純粋自然科学サークルだったのですね。相対論分科会とか量子論分科会とかしていたようです。平安京エイリアンの頃

「理論科学グループがコンピュータなんかしていて良いのか！」

という意見があったなんて信じられませんね。

とにかく現在は理論科学グループはコンピュータサークルです。これからもそうでしょう。コンピュータの社会における役割は、これからさらに重要になってくることは言うまでもありません。もはや

「理論科学グループだからこそコンピュータを極めるのである！」

というべき時代でしょう。

近年、コンピュータの普及はめざましく、特に情報教育棟の開設により猫もしゃくしもX端末という状態になりました。情報棟ではネットワークプログラムがすぐ作れてしまうのです。そもそも数値計算のために生まれたコンピュータが、これからは人々の間のコミュニュケーションを促し、たくさんの幸せを運んでくれるでしょう。さらには、人々の力がネットで合わさって世界規模で事を動かすようになるでしょう。そんなプログラムを自分で作れてしまうのです。この環境は非常にありがたいものです。プログラムを作って一番うれしいことは、その利用者が喜んでくれることです。その喜びがすぐに返ってくるのが、情報棟の楽しいところです。

しかし情報棟にはやはり低俗なユーザーが増えてしまっています。駒場生全員にメールを送ろうとして150kbyteのheaderのメールを作りあげたおばかさんたちはまだ良いほうです。ワークステーションでゲームをする連中も嫌いですね。もったいないし邪魔ですし、みっともないし、東大生なのですから情けないです。いたずらメールを作る奴とかパスワードを盗もうとする奴がいるというのが悲しいです。

コンピュータの健全なるエキスパートであるTSGの会員はこの駒場の情報棟の安全な環境の保持のために、ユーザー達を指導していかなければなりません。

でも情報棟には低俗な人ばかりではありません。プログラムを作りたいという希望を持つ若いきらきらした芽もたくさんあるのです。彼らの希望の実現を助けることは、将来のコンピュータ社会に明るい光をあたえることにもなるのです。このような若い芽を育てていくこともこれからのTSGの役目と思います。

もはやコンピュータは一部のマニアのものではありません。TSGは一部のプロフェッショナルの集いの場であることには変わりありませんがそれだけでなく、広く一般の人々にコンピュータの意義を教え、その有効な利用を指導していくことも必要になってきているのです。

時代は明らかに変わったのです。

TSGの部長に本当に必要なことは、プログラム技術力でもなく、会議を欠席しない勤勉さでもなく、それは一般の人々によるコンピュータ社会の健全な発育を目指す精神と、その精神を伝える布教力なのです。

でもね。なにかに言っても、やっぱりTSGはサークルなんだから、みんなが楽しく過ごせればそれで十分なんだよね。別にみんなが一丸となってコンテストに出る必要もないのだよ。わいわいしながらお弁当を食べるという長閑かな環境でいられればそれでいいんだよね。

僕も前の部長から言われたけれど、楽しければいいんだ、ということを強く思うね。
というわけで、次の部長さんは頑張ってくださいね。
では。

理論科学グループ根津中央研究所所長　渡辺 尚貴

'95 編集長　　ちょもらんま

編集長の仕事は、仕事のあとが形になって残るのがいいですね。「ああ、オレはこんなによく働いたんだなー」みたいに実感が沸きます。

部報に関しては、179 号で HRD さんが書いていた編集部への要望を実現させるのが、僕の目標でした。発行回数・ページ数・カラー化など、結構納得のいくものになったと思います。(注：カラー化に関しては、190 号の赤帯があります。(^^;)　各号を振り返ってみましょう。

■ 186 号（新旧役員挨拶号）

発行	:	1994 年 12 月 17 日
表紙	:	サンタとクリスマスツリーのイラスト
色	:	緑
ページ数	:	24p
なお率	:	13.5 %

初めての部報で、最も苦労した部報でもあります。確か、締切りを守ってくれたのは 4 人しかいなくて、最後の原稿が入ったのなんか発行日当日の 8 時過ぎでした。もちろん発行前夜は一睡もする時間がなく、原稿集めの厳しさを感じました。(^^;

発行前日、原稿が少なくて死にかけているときに、Nishi が 8 ページもの原稿をもってきてくれました。部報と呼べるだけの厚みに出来たのは、彼のおかげです。あれは本当に天の助けでした。

慣れない作業なので苦労したという面もあります。「編集長になった、ワープロが必要だ」ってときにたまたま親父の FM-OASYS しかなかったというだけの理由で OASYS を使って編集したのですが、こいつが DOS のソフトと使い勝手が全く違うので、苦労しました。「いつかは他のものに乗り換えよう」と思っていたのですが、慣れるとそう悪くないので、結局最後まで OASYS で通してしまいました。

■ 187号（スキー合宿号）

発行　　：1995年1月12日
表紙　　：スキーヤーのイラスト
色　　：オレンジ
ページ数：26p
なお率　：30.1％

186号と同じくらい苦しかった部報です。まだ原稿集めのなんたるかを知らなかったということでしょう。(^^; 発行2日前の時点で原稿がほとんどないなんて、後にも先にもこのときだけです。186号ではNishiに救われましたが、このときはNAOに救われました。自分でも、必死になって書きました。あのころは元気だったんですねえ。

名言・迷言集は、ずいぶん好評だったようです。かなりきわどいことをたくさん書いちゃいました。あじさん、うんぶさん辺りに殺されるのではと心配していたのですが、ちょっと苦情を言われたくらいで済んでよかったです。(^^;;

■ 188号（追い出しコンパ号）

発行　　：1995年3月12日
表紙　　：だるまのイラスト
色　　：黄緑＆水色
ページ数：30p
なお率　：29.3％

とにかく原稿をたくさん集めたいので、顔を見たこともない上の代の先輩方に頼みました。本当は原稿と言うよりは数行程度書いてもらって寄せ書きみたいにするつもりだったのですが、上の代の人達は律儀に原稿を書いて下さってので、とてもありがたかったです。それなのに、お名前の漢字を間違えた私は、なんてバカなんでしょう。(;_;) NAOにフォローしてもらったのですが、配ったあとにもう一人間違えていたことが判明してしまったのでした。(T_T)

NAOと小島さんの原稿が入っていますが、2人とも確実に締切りを守ってくれるので、ありがたかったです。TEAさんの「ごみ＊．LZH」は、この号が最初でした。最後の8ページほどに何かあやしいのがありますが、恥ずかしいので捨てちゃってください。

編集後記を自分だけでなく原稿執筆者にも書いてもらうようにしたのは、この号が最初です。一人で箱に閉じ込められているのは寂しいもんね。

追い出しコンパ来た方が大量にもらっていったらしく、50部刷ったのにもらえない方が出てしました。前代未聞（空前絶後にはなりませんでしたが）の再発行をするハメになり、30部増刷しました。

■ 189 号（オリパンフ号）

発行　　：1995 年 4 月 12 日
表紙　　：宝船のイラスト
色　　：水色、黄緑、クリーム、菫色[1)]
ページ数：56 p
なお率　：38.3 %

今期最大の原稿量を誇る号です。縮小印刷しないと紙が足りなくなるという快挙を成し遂げました。最大の功労者はもちろん「恐怖の大量原稿送信」とか言って 45KB の原稿を送ってきた NAO ですが、あの字の小ささにも係わらず 11 ページ占領している Zephyr さんの原稿の凄さも忘れてはいけません（分量だけでなく、内容もキてましたね）。

苦労話は、ほとんど 190 号に書いてしまいました。あのときは書き忘れましたが、P7 に載った「いぬ。BBS」の電話番号は物議を醸しました。SYSOP が原稿に電話番号を書いたのだから載せて構わないだろうと思ったのですが、印刷したあとで結構反対意見が出てしまいました。インクが潰れて電話番号がよく読めなかったので、心配することはなかったのですが。僕としてはいぬ。の存在を早めに 1 年生の目に触れるところに置こうと思って ったく☆ さんに原稿依頼したのですが、不特定多数の人に配るオリパンフ号よりも、新入生自己紹介号に載せた方がよかったかも知れません。

この号からプリンタが MJ-5000C に変わりました。もうひとつ、ゲスプリに「写真モード」なるものがあるのを発見。普通に印刷するより格段にディテールが綺麗になる上、濃淡の表現まででできることがわかったのです。(^^) 第 1 刷を持っている方は、P31 の一番上を見てください。鉛筆と消しゴムのあとが見事に再現されています。

「きょーふの原稿依頼^^;」が乱れ飛ぶようになったのも、この頃です。

■ 190 号（新入生自己紹介号）

発行　　：1995 年 5 月 13 日
表紙　　：6809 のアセンブラのアセンブラソース
色　　：黄色
ページ数：80 p
なお率　：31.8 %

なんか知らないけど、やたらと分厚くなった号です。それまで部室の本棚で発見した最も厚い部報は 130 号（1987 年発行）の 76 ページだったのですが、編集しているうちに 74 ページに達したので、部室に現存する部報の最高ページ数記録を塗り変えるのは今しかないと決心しました。(^^; NAO と小島さんに急遽追加の原稿依頼をして、自分でもくだらないことを書いて 80 ページの大台に載せました。まったく恐ろしい話です。

記録樹立の記念に表紙に赤帯を入れたのですが、これは結構大変でした。75 枚にカラー

で印刷するのは、予想以上に時間のかかる作業でした。おかげで部報の製本作業が遅れ、コンパ委員に迷惑をかけてしまいました。新歓コンパで人が集まってなければ、製本は間に合わなかったことでしょう。手伝ってくれたみなさん、どうもありがとうございました。

189 号までの表紙はTOWNS システムソフトウェアのイラスト集でごまかしていたのですが、オリパンフを作るときEIN さんに「表紙もかっこよくね。」と言われてしまったので、この号は凝ってみました。大昔に書いた6809 用手製アセンブラのソースを印刷して、カット・ペースト＆コピー（注：鋏、糊、10 円コピー）で仕上げました。「あやしいアセンブラ」と言う人はたくさんいたのですが、6809 だと気づいてくれたのあじさんだけだったみたいです。あじさん曰く、「このアセンブラ、アドレッシングの表記がちょっと違うね。」う〜ん、流石。

編集スタイルもほぼ固まってきました。186 号などとは比べるべくもないほど綺麗に仕上がっています。

■ 191 号（夏合宿前号）

発行	:	1995 年7月3日
表紙	:	八丈島の底土港に入港する小笠原丸
色	:	ピンク
ページ数	:	52ｐ
なお率	:	68.6 %

取り込み写真を多用した号です。個人的にゲスプリを使ったときに、写真でもかなり綺麗に印刷ができることは確認済だったので、使いそうな映像を手当たり次第ビデオで撮影してあちこちに散りばめました。

表紙に使う写真を探すのは大変でした。小学生の頃に小笠原に行ったことがあるので、そのときのビデオテープから適当に選ぶつもりだったのですが、実際にテープを見てみると、表紙に耐えるようなシーンが全然ないのです。小さい頃の自分が映っているところは恥ずかしいし。何時間もあるテープをずーっと早送りしながら表紙の候補になるなシーンをディジタイズしていったのですが、ただでさえカメラワークの下手な映像を早送りしながら1時間以上見ていたので、目が回って気持ち悪くなり、合宿前から船酔いの気分でした。でも、苦労の甲斐あってか、選んだ画像は大島の雰囲気と結構合ってたみたいです。

他の原稿が少なかったので、NAO 率がついに過半数を突破しました。182 号（今の2年が自己紹介をした号）で、箇条書きで簡潔にキメていたころのNAO の面影は、もうどこにもありません。(^^;

■ 192 号（夏合宿号）

発行　　：1995 年 9 月 7 日
表紙　　：夏合宿の集合写真
色　　：あさぎ色
ページ数：74p
なお率　：15.9 %

夏合宿の原稿をあちこちに依頼していたら、合宿ネタだけで 30 ページを突破。合宿で取った写真をそのまま貼ったり、サイズが合わないものは情報棟のスキャナで取り込んだりして使いました。P8 のリスの写真は、ゲスプリでここまでできるのかと思わせるほど綺麗に印刷できました。表紙のロゴは、お約束通り野田浜海岸の砂浜で作った砂の「TSG」を使いました。

合宿ネタもさることながら、Nishi の「超々爆速 lh5 生成法」と TARO さんの「Jerusalem Virus 解析結果報告書」は圧巻でした。特にウィルスの原稿の分量はすさまじく、NAO の C の記事が無いにも関わらず 74 ページもの厚さになってしまいました。

この号を作るときに、とんでもないトラブルが発生しました。発行日前夜にプリンタのインクが切れて、印刷できなくなってしまったのです。仕方なく、写真や没印刷を切り貼りしたり、手書きで修正してごまかしました。インクがあれば、目次の「7」と「9」は、まともなフォントになっていたはずです。

■ 193 号（駒祭前号）

発行　　：1995 年 9 月 7 日
表紙　　：平安京エイリアン
色　　：紫
ページ数：54p
なお率　：59.4 %

GANA の平安京エイリアンの原稿が面白いですね。表紙も GANA の原稿の絵をそのまま使わしてもらいました。ったく☆さんの原稿も入っています。NAO の原稿は相変わらず凄いです。

編集のスタイルは 193 号をそのまま踏襲しました。

■ 194 号（駒祭号）

発行　　：　1995 年 9 月 7 日
表紙　　：　十二支
色　　：　紫、水色、オレンジ、黄緑
ページ数　：　10 p
なお率　　：　40.2 %

一般向けということで、薄く・読みやすい部報を作りました。この試みは成功したと思っています。おおいわ君の四柱推命やＮＡＯの理論科学シミュレーションの原稿は、短く決まっていて読んで気持ちがいいです。Makken さんの原稿は、もらったときはなんだかわけがわからなかったのですが、エヴァンゲリオンのパクりだったわけですね。

表紙の十二支は、1 年の松永さんが書いてくれました。

■ 195 号（駒祭反省号）

発行　　：　1995 年 12 月 16 日
表紙　　：　？
色　　：　？
ページ数　：　？ p
なお率　　：　？ %

編集どころではありません。早くこの原稿を書き上げなければ……

■ 名簿

やたらとスタートが遅かったです。9 月には始めた方がいいでしょう。授業が始まるまで放っておくと、あとで身動きがとれなくなります。僕は 10 月後半に始めたのですが、かなりやばかったので、編集を副編集長の岡村君に任せて情報集めに徹しました。

メールを出せば大半の人はちゃんと名簿のデータをくれるのですが、情報棟に慣れていない僕はメールを出すだけでも結構苦労してしまいました。メールアドレスが無効の人も多く、何十人も電話攻撃する必要がありました。学部生はともかく、教官の方にまで電話するのは、気が重い仕事でした。

メールや印刷では、GANA に随分助けてもらいました。TEX のソースは、HRD さんの'93 名簿のソースを流用しています。

そうそう、シリアルナンバーをまだ申告してない人は、僕まで知らせてください。

■ 最後に

部報月刊化は達成できませんでしたが、計10回の発行はまずまずだと思います。仕事を増やして、自分の首を締めていただけのような気もしなくはないですが。この発行ペースを維持しながら分厚い部報をコンスタントに発行できたのは、原稿執筆者の主力だった2年生が締切りを守り、まとまった量の原稿をくれたおかげです。特にNAOには随分書いてもらいました。いいときに編集長をやらせてもらったと思っています。一冊のページ数が抜かれることはあっても、総ページ数の記録は不動のものになるでしょう。

あと、編集長としての仕事以外にもいろいろとやりました。NAOが実験のときに、代わりに学友会の総会に出たこともありました。今年度は、「きゃんぱそ」、「週間ファミコン通信」、「コンプティーク」と3回大手のパソコン誌に載る機会がありましたが、なぜかたまたま電話を受けたのが僕だったりして、3回とも担当者になってしまいました。ファミ通の方に聞いたのですが、僕は「外報部長」とかあやしげな称号を賜っていたようです。

'96年度には、第200号という大きな節目があります（新歓号かな？）。おおいわ君がどんな部報を作るのか、楽しみです。

■ おまけ　〜原稿のファイル名〜

186号	新コンパ委員・庶務係	Yu	TAWAKE.DOC
190号	たてかん	GANA	**タテカン**GANA.TXT
190号	新入生自己紹介	松村	**自己紹介.毒**

æ

'95 ライブラリアン　gana

昨年は「月に 100M 増える MO のデータは、手に負えない。」と一言ありましたが、今年はあまり増えず、そのうえ 305 ウィルス事件の後には №0 が事実上の 486GR の支配者となり、僕は何もやってない状況でした。

しかし、めでたく

- TSG home page の管理・運営
- Circle's home page の一覧 (TSG 版) の管理・運営
- TSG home page に部報を掲載する

という新たなる仕事ができましたので、GEL はがんばってください。

きっと home page のデータは代々受け継がれ、ふくれあがり、相談員となることが必須になることでしょう。

'95 会計 Nishi

あっという間に 1 年が過ぎてしまいました。振り返ってみるとそんなに大変ではなかったですね。あえて言うなら、買い出しに誰も付き合ってくれなかったことと、486GR 上の表計算ソフトを使って会計処理をしていたため、305 ウイルス事件のとばっちりを受けたことぐらいでしょうか。

まず、私が会計をしているときに買った物ですが

- 15 インチマルチスキャンディスプレイ
- CD-ROM ドライブ
- MO20 枚（ぐらい）

その他、文房具、また印刷費、紙代等の支出がありました。

（具体的な金額については、私の中間試験・レポート提出が終わるまで待ってください。）

ディスプレイは去年駒場祭のときにそれまでのディスプレイがお亡くなりになり、HRD さんのを急遽借りました。この借り物を使いつづけるのも良くない、という訳で購入したのですが、HRD さんのディスプレイはその後もずっと 305 に置いてあったような … 。まあマルチスキャンなので（それまではマルチスキャンではなかった）良しとしましょう。でもこのディスプレイ、安いだけあって不便な点もありますね。上下に首を振れない、ってのがそうです。長時間使っているとこちらの首が痛くなります。

CD-ROM は NEC ドライブにしておいて正解でしたね。実は同じ値段で高機能の AIWA の倍速ドライブも検討していたのですが、将来を考えて台数の出ている NEC 製にしたのです。こちらにはケーブルが付いていなかったのでその分高く付きましたが、AIWA ドライブは最近その接続性に色々と問題が生じているようです。

また MO は 10 枚一気に購入したのですが、当時は 128M の MO でも 1 枚 1000 円以上しました。MO の値段の下がり具合を見ると、これは失敗だったかもしれません。

まあ基本的には支出を抑制しました。大赤字状態の TSG 会計を考えるともう 1 年くらい支出を抑制すべきかもしれません。（大赤字状態については新会計の人に聞きましょう）

なお、305 にある扇風機・プレステは有志による基金によって購入された物です。

それから会計の仕事ですが、実に簡単ですね。基本的には仕事はこの 2 つです。

1. 3月31日までに予算を使う。

2. 領収書をノートにぺたぺたと貼って、会計簿をまとめ学友会に提出する。

1に関してですが、TSGには3月（それも31日ごろ）に買い出し行くという伝統があるようです（私は29日に一人で買い出しに行きました。）。今度はそんなことの無いように早めに話し合って買う物を決めておきましょう。

あと2に関してですが、ただし書きが明確でない領収書、宛て名が無かったり、上様になっている領収書、あと生協・コンビニ等のレシートを学友会は領収書として認めてくれません。皆さん気を付けましょう。また会計簿の提出は毎年5月の連休明けごろにあります（確認しておくよーに）。提出が近づいたら会計担当に「かいけー、かいけー。」とか言って注意を喚起してあげましょう。そうでなくても会計簿は早めに整理しておいた方が無難です。提出期限寸前にウイルスの影響を受けてパニックに陥る事があります。（しつこい）

最後になりますが、金品の管理には十分気を付けましょう。もちろん使いすぎにも。

'95 学館連絡員 Nishi

学館連絡会議の存在を忘れないよーに。掲示を見つけたら学館連絡委員に教えてあげましょう。

'95 コンパ委員 しぐま

宴会の支度に追われつつの、あっと言う間の一年でした。

いきあたりばったり[2] あり、アクシデント[3] ありで、色々とご迷惑もお掛けしましたが、私自身は、とても楽しく困憊員^H^H^Hコンパ委員を務めさせて頂きました。

最後に、次代のコンパ委員の為に。

`tecc.circle.tsg` を読める方は読んでください！

では、1年間、おつきあい頂きどうも有難うございました。

[2]「宿の場所、どの辺り？」「宿の人が連れて行ってくれるまでわからない ···· 」於夏合宿
[3]某コンパで、予約した1件目の店は潰れ、2件目の店がいきなり改装をする事になった。

'95 コンパ委員　　しげ

95 年度コンパ委員のしげです。やっと仕事が終わって一安心というところですが、今年は実質コンパ委員が 2 人だったのに（ひどすぎかも）楽をできたのは何と言っても「史上最強のコンパ委員」の異名を取った sigma さんのおかげでしょう。基本的に店の予約までは完全に彼女がやってくれました。僕も新歓コンパ一回だけ予約しましたが何故か（店の責任だとは思うが）予約したことになっていなくて大変な思いをしました。そのときは同じくらいの人数の集団を別の店に移してもらうことにより解決したのですが、その集団は TSG よりも高い料理を予約していたらしくて大変でした。

そこで新コンパ委員に一言。基本的に一回のコンパは一人のコンパ委員が予約からお金の支払いまでしきるか，少なくともどういう条件になっているかを予約した人から詳しく聞いておくこと。店にだまされることがよくあります（まじで）。他のコンパに関してはせいぜい最後にお金を徴収するくらいでした。その上，試験のため夏合宿にも参加できず本当に sigma さんには頭が下がります。結局僕がやったことで一番貢献したと思えるのは，PlayStation 基金なるものを作って 305 に PlayStation を導入したことでしょう（うんうん）。

おまけ ～副部長編～

また同時に僕は副部長でもあったわけだが，こちらの方は一回も仕事をしていないと言って過言でない。そのため僕が副部長であったことを知らない人もたくさんいたと思う。これは非常にいい傾向である。どうしてかというと副部長は部長をサポートするための役職だからである。つまり部長がしっかりと仕事をしてさえいれば副部長は仕事がないのである。いわば部長が使えない奴だったときの保険だね。ご存じのように今年の部長はなおという非常に働き者だったので僕の出る幕はなかったのである。

あーよかった。

ま、しかしこんなにしっかりした人が部長になるというのも珍しいので(^^;「何月何日に会議があるよ」と部長に教えてあげるくらいはしないといけないかも。

'95 コンパ委員　Yu

じつはコンパ委員だった渡邉です。私の顔を知っている上の人はいったい何人いるのだろう？

TSGではコンパに出席しないと上の人と顔を会わせる機会がないようなので、私の事を知っている人はほとんどいないでしょう。私も上の人の顔と名前はあんまり一致しません（ひどいかも）

就任の挨拶で私のモットーについて書きました。結局わたしはモットーを貫いたわけですが、そのせいで他のコンパ委員二人に多大なる迷惑をおかけしました。この場を借りて一応おわびしておきます。まぁ、私が働いても事態はそう変わらなかったと思いますが。

コンパ委員の心得うんぬんについては私は何も語ることはできないので他の二人に聞いてください。結局わたしはいったい何だったんでしょう（爆）

'95 庶務　Yu

いやぁ、いつのまにか1年たってしまいましたねぇ。就任当時はどうなることかと思いましたがどうにかなってしまいました。これも全て皆さんのおかげです。御協力くださった方々、どうもありがとうございました。

さて、庶務の仕事と言うのは非常に大変です。何しろ無法地帯ともいうべき305の掃除がその主な仕事なのですから。どのくらい大変なのかと言うと一人ではとても出来ないので思わず部長に仕事を頼んでしまうほどです（笑）

今年は一人では大変だろうと言う後輩思いの私の意見により、庶務は二人に拡張されました。二人いれば他の人の応援を頼むことなく仕事が出来るだろうと思いましたので。きっと今年はごみ箱がいっぱいになって一週間以上放置されるということもなくなるでしょう。

最後に、一言。空き缶はくずかごに（特に某氏）

TSG新役員就任挨拶

'96 部長　わたる

96年度のTSG部長になった わたる です。どうして私が部長に選ばれたのかは、実は全くの>>謎<<なのです。部室305の常駐率は高くないですし、前部長のように『理論科学』に強いわけでもなく、怒涛の大量原稿も書けません。NAOさんとの共通点は通学時間が往復で3時間半であることくらいでしょうか。しかし、もちろん責任はちゃんと果たすつもりですのでご安心を。(とか言いつつ、先日は自治会のサークル代表者会議をいきなりすっぽかしたのであった(^^;)

新部長として、TSGの今後の活動について考えていることを述べさせて下さい。私はプログラマーを育成する環境を整備することが必要だと思っています。TSGに入る前からプログラムを作れる人は年々少なくなると予想しているからです。なぜなら世の中にはWindowsが蔓延り、ソフト開発に必要な資料やコンパイラはとても高価で、プログラマーの負担は増大しています。その一方で見栄えの良いグラフィックを使ったゲーム機が普及しており、それらと比較しても満足できるソフトを素人が作ることはとても難しくなっています。つまり世間にパソコンはますます浸透していくでしょうが、プログラミングのできる人が増えるわけではなく、むしろ逆に減ってしまうのではないかというのが私の危惧なのです。だからTSG内に全くの初心者でもCなりマシン語なりを勉強して、ちゃんと動くプログラムを作れるようになれる環境を整備したいのです。さもないと私たちのサークルは今年や来年は大丈夫でも、将来的には今まで受け継いできたような活動ができなくなるかもしれません。今年についてさえ、駒場祭の四柱推命のプログラム部分はaleph-0君一人に依存してしまい、占いの印刷部分を凝るところまで手が回せず、せっかく用意した十二支の絵が無駄になる有り様でした。

それで具体的な対策ですが、分科会の活動をもっと積極的にするくらいしか、今の所は情けないことに思いつきません。C言語の勉強会については、他の自然科学系サークルと連携したインターサークルゼミをやろうという提案が前部長からありまして、これはさっそく開始になりました。プログラミングは「習うより慣れろ」なので、次の駒場祭ではゲームを共同制作して展示するようにするという案もあります。しかし開発効率はプログラマの人数の2乗に反比例すると言う説もあるくらいですし、なかなか難しそうです。

長々と偉そうな感じで書いてしまいました。私はただでさえ未熟者であるのに大ボケ

な性格まで持っていますので、迷惑をかけてしまうこともあるかも知れません。皆さんの協力と助力が必要ですので、今後ともよろしくお願いします。

最後に私の開発環境などを紹介します。所有マシンはPC-9821で、言語はCとマシン語、ほんの少しだけC++を使います。マシン語についてはプロテクトモードだけで、リアルモードはよく知らないという変なやつです。GNUのDOSエクステンダGO32上で動作するソフトをdjgppで作っています。djgppは基本的にAT互換機用ですので、98で使うには色々と苦労がありますが32ビット環境は得るものが大きいです。互換機用のグラフィックライブラリを98に移植して、ベクターデザインのPACKをもらえる身分になろうと言う計画(^^;を副部長と現在進めているところです。

'96 副部長・学館連絡員　　高野 直樹

副部長に任命された高野です。

ついでに学館連絡員にもなっています。駒祭では四柱水命のテキスト打ち込みとフライトシュミレーターのお手伝いをちょこっといたしました。ついでに呼び子もやってました。(けっこうでかい声を出すのが得意でカラオケではシャウトしたりする。)ただいまC＋＋を勉強中です。とりあえず今後の抱負を語ってみたいと思います。

「とりあえずC++を勉強し終わったら何か作る。」 「何か」というのがみそなのですが、今の所gccのLIBの移植と3Dのプログラムをくんでみようかと思っています。とりあえず4月のオリエンテーションまでにはなんか一つは完成したいです。(ホントウカ)

「駒祭を成功させる。」 今年の占いは他のサークルの占いにだいぶ客を取られてしまったらしいので、なんらかの対策をたてたい...(来年も呼び子をやるのはもういやだ)

けれども来年の新入生にはプログラマーがいるのだろうか？ということで...

「プログラマーを育成する？」 私自身がたいしたものも作れんのにこんな大口をたたいてしまってよいものか？

その他の抱負としては

「ODPとハードディスクを買う。」 さすがにこのまま(486SX-25, HDD300)では辛くなってきた。家庭教師でお金を稼ごう。ただIDEがエンハンストではないのでSCSIを買う羽目になりそう(;_;)

しかもODPもCe2は基盤ごと取り替えるのでめちゃくちゃたかいし(;_;)

（今時 486DX2 で何で 4 万もするんだ？（怒））

「年間 200 冊 !?」 漫画を年間 200 冊集めるのはさすがに無謀だな。せめて 100 冊は....

（最近は羅川真里茂にはまっていたりする。）

「テニスとスキーとゲームをやる」 実は全部苦手だったりする。特にスキーはまだ 2 回しか行ったことがない。スキー合宿で遭難しそうになったら助けてください。

これ以上書くと何かぼろがでそうなんで.....
それでは 1 年間よろしくお願いします。

'96 編集長 Aleph-NULL

今回編集長を継ぐことになった Aleph-NULL ($\aleph_0$) こと 大岩（おおいわ） 寛（ゆたか） です。

とりあえず、原稿が揃えば月 1 回の発行を目標に頑張る所存です。ちょもらさんの時は、毎号 70 ページを突破すると言うなかなかすごい状態になっていましたが、今年がどうなるかは原稿の集まり方次第です。

なお今回は、部報の編集は LaTeX を使う予定です。TeX は理科系の人間には一番使いでのあるソフトでしょう。皆さん理科系の教養として TeX を覚えましょう。

あと、TSG を TSG as Tennis, Ski & Game でなく TSG as Theoretical Science Group とすべく、正式な分科会を作りたいと思ってます。とりあえず C 言語演習分科会、Textfile 分科会でもやりたいと思ってます。あと誌上 TeX 分科会なんてのもやりたいけど原稿を書く暇があるかなぁ。

原稿を募集しています。投稿の形式は次の通りです。

形式

プレーンテキスト (MS-DOS/UNIX, EUC/SJIS/JIS)

LaTeX の原稿ファイル (`\documentstyle[b5j,`*ascmac*`]{jarticle}`)

原稿の提出先：

いぬ。のアドレスが変わっています。

g540001@komaba.ecc.u-tokyo.ac.jp

OIWA@**いぬ。**BBS

なお、原稿依頼がありました際は、ぜひ進んで御執筆くださるようお願い申し上げます(^_^)。

'96 会計　村井 源

なぜだかしらないけれど会計にされてしまった村井です。

中高通じて柔道部だったので脳味噌まで筋肉に浸食されており足し算とかの難しいことは出来ません。しかしここには星野君という非常に会計向きの方がおられるので実質上の会計はたぶん彼でしょう。

もう一つはいっているサークルの方でも何か知らないけれど役職をつけられてしまったので、多少心配な今日この頃です。

買い物するときは暇な人はきてください。ぼくは方向音痴なので一人でいったら帰ってこれなくなるかも知れません。それにDOS/Vのことはさっぱりわからないのでぼくに任せるとどんなマシンを買ってくるか保証できません。

こんなぼくですがどうぞよろしくお願いします。æ

'96 コンパ委員　かもしま

僕が此の度コンパ委員に就任した鴨島です。

コンパ委員と云われても僕は全然店とかを知らないのではっきり云って困ってます。今の時点ではクリスマスコンパの店すら決まってません。此の号が出る迄には決まっている筈ですが、果たして大丈夫なんでしょうか :-)

かくの如く心許ない状態なので前任者の方々に色々お尋ねしながら何とか職務を遂行していこうと思っています。宜しくお願いします。

'96 コンパ委員　窓明

新コンパ委員になったのむら（窓明）です。

今年は、TSG内の遊び人をまとめて4人コンパ委員にしたようで、なんか実務能力に欠けているような気もしますが、温かく見守ってやってください。

しばらくは、「コンパ委員マニュアル」なるものに頼ることになるでしょう。これほどまとまったコンパ用マニュアルも珍しいのではないでしょうか？

前任者の方々に感謝 !!

関西圏の人間のため、あまりコンパ会場に詳しくないのでその手の情報に強い方は、ぜひ教えてください。

それでは、「テニスとスキーの、自然を冒涜するサークル」のコンパ係として一年間よろしくお願いします。

……なお、私は「まんがくらぶ」の部長さんだったりもして、駒祭前後は忙しくなりそうなんでそこは他の３人の健闘を祈っています。(^^;) **ヒドイ**

'96 コンパ委員　ZERO

何故かコンパ委員のZEROです。はて、コンパは新歓コンパと駒祭の打ち上げしか出ていないような気がするのですが... ま、細かいことは気にしない、気にしない... コンパ委員は4人ともプレステ人間という話もありますが、そんなことは気にしない、気にしない...

というわけで、余り物の役には立たないかも知れませんが、他の3人の人々に迷惑をかけないよう頑張りたいと思います。コンパでは主に2次会参加要員でしょう(^^;

物の役に立たない分、村井君（会計）の足し算の検算(^^; や，げる（ライブラリアン）の仕事の一部を少しは手伝おうかなぁ，と思う今日この頃です。

みなさんには迷惑をかけないよう（クリスマスコンパの二の舞にならないよう(^^;) 頑張りたいと思います。1年間宜しくお願い致します。

'96 コンパ委員　　きりもみ

世の中なにが間違っているかというと、これだ。私になにかをさせようなぞ、間違いとしか言いようがない。これからは、鉄拳の強さで決まることに決まったようだ。というわけで、鉄拳について語りたい。

まず、鉄拳は、最初人気がなかった。高校の夏頃、初めてゲーセンで見た鉄拳は、相当人気がなかった。取っつきにくいのがネックとなり、みんなやらなかった。僕も一八で平八まで行ったが、クリアはしないまま時は過ぎた。

で、PS を買った。鉄拳が出る。一応購入だ。しかし、いざ腰を据えてやると面白い。風神拳が出るようになって、初めて楽しさを知ったようなものだ。

空中コンボも出るようになり、ゲーセンへ突入だ。しかし、強い人には勝てなかった。富山なぞという田舎と違って、あまりマークされていなかった鉄拳をアーケード版発売と同時にやり込んでいたとおぼしき毒人間たちのひしめく街東京。僕は感動と同時に、あきれた。富山の人はそのころスト 2X をやっていたのに …。(ちなみに富山は保守的で、ヴァーチャもやらない)

すでに部室では、鉄拳は社会現象だ。みんな昼休みには、空中コンボの練習。もめ事も鉄拳で解決。M 君の玉子ボーロがなくなってもめたときも、鉄拳で解決したのは記憶に新しい。

というわけで、来年新入生になる人たち、などは、みんなのあこがれのコンパ委員になるために、勉強はそこそこにして、鉄拳をやれ。しかし、この文章を今読んでいる人は来年新入生になる確率が皆無なところがつらいところだ。

ともかく来年は、コンパ委員らしく、鉄拳分科会なども行いたい。尚、3月からはおそらく鉄拳 2 による講義が主になるだろう。2 での最強空中コンボも募集中である。みんな頭をひねって考えて、おなかを壊すように。

これからはマスコミにも働きかけ、鉄拳最強グループ(TSG)の名をとどろかせましょう。

'96 庶務　はと

'96 庶務のはとです
庶務ってなにするんでしょう？ 305 の掃除以外のことは聞いてないので

「みなさん 部室をちらかさないように」

とかここで言っておけばお仕事は終わりだな ^^;
先学期はあんまし部室に行ってなかったんですが、今学期は多少向上心が出たのでいろいろ教わりに行くと思います。それから、僕は TOWNS ユーザーなので、部室で死んでる TOWNS を使ってみたい人は言ってみてください。(でも 弱い TOWNS ユーザーですのでご了承下さい)

'96 庶務　竹島 秀則

庶務係の TAKE （竹島 秀則）です。主に掃除をする係なので、掃除します。あとは知りません。まだ自己紹介をしていないので、ここで自己紹介させて頂きます。

クラス:	理科一類 1 年 19 組
出身高校:	桐蔭学園高校（神奈川県）
e-mail アドレス:	g540742@komaba.ecc.u-tokyo.ac.jp
所有機種:	PC-9801BA3/U2
	HDD 210M, MS-DOS 6.2, Windows 3.1

パソコンを触ったのは、今年 4 月が初めてです。というわけで、パソコンについては、よくわかりません。
一応、多少は C 言語を使えます。（まだ勉強中）でも、パソコンは現在、ゲーム機に近い状態です。
その他、ファミコンとスーファミがあります。昔は、スクウェアの FF シリーズやサガをやっていました。最近は、マザー 2 をやっています。（まだクリアしていません）
とりあえず 5 行を越えたので自己紹介を終わりにします。æ

XToolkit intrinsics

GANA

1 XToolkit って一体 · · ·

The UNIX Super Text (下) より

> widget という概念を導入することによって, クライアントの作成を容易にすることを図ったライブラリです. widget とは, 押しボタンやメニューといったクライアントのユーザーインターフェースを構成する部品のようなものと考えてください. XToolkit を用いる場合, この部品を組合せ, その動作を調整することにより, クライアントを作成していきます.
>
> XToolkit の中にはそのまま利用できる widget は入っていません. XToolkit を用いてクライアントを作成する場合には, XToolkit とは別に Widget セットのライブラリをリンクする必要があります.
>
> X ウィンドウシステムでは標準的な widget セットライブラリとして, Athena widget セットを用意しています.
>
> (略)
>
> widget はオブジェクト指向の考え方から設計されたもので, クラスという概念と継承という概念が導入されています. (略) Xtoolkit を使ってクライアントを作成する場合, 必要な部品のクラスのインスタンスを生成し, そのインスタンスを組み合わせてクライアントとします. ちょうど, タイヤというクラスのインスタンスであるフロントタイヤ, リアタイヤを作り, これらを組み合わせてバイクを作ると言ったようなものです.

だそうですが, 以下の説明では widget セットは利用しません. Athena Widget や Motif や OPEN LOOK を使うことが目的ではないのです. 又, widget のクラスを自分で作成したりもしません. 継承も考えません. これからやろうとしていることは, 「あくまで Xlib がメイン. でも XToolkit の便利な所だけは利用してしまえ〜」 ということなのです. よって, XToolkit の一番基本的な部分である XToolkit intrinsics と, 基本的な 3 つの widget class しか使わないことになります.

2 よいところ

- コマンドラインの自動解析
- リソースを利用することによりユーザーカスタマイズを容易にする
- イベントドリブンのプログラムをより書き易くする

などです.

コマンドラインの自動解析は説明するまでもありませんね. 自分で解析するのは結構面倒ですから. それから, -geometry や -iconic や -fg や -bg などなどのオプションをディフォルトで解釈してくれるようになります.

リソースの利用については, 例えば, ~/.Xresources (人によっては ~/.Xdefaults かもしれません) に次のような記述をしたとしましょう.

```
*Foreground: White
*Background: Red
```

すると, XToolkit を利用していれば, window の背景色が赤になり, 前景色が白になります. 又, キーカスタマイズも非常に簡単に設定できるようになります.

イベントドリブンのプログラムが書き易くなるのは, あるイベントが起こった時に XToolkit がそれを処理する関数を直接呼んでくれる為です. Xlib だけを利用していたとしたら, イベントごとに switch などで自分で振り分けなければなりません.

そうそう, XToolkit とは関係ありませんが shape window の作り方についても少しだけ説明することにします. shape window とは, oclock や, xeyes で使用されている矩形でないウィンドウのことです.

3 リソースって何だ?

widget はそれぞれの widget class に応じて色々な属性を持っています. 例えば, button と書いてあるボタンの widget では背景色が黄色で文字の色 (前景色) が紫で文字列が button でフォントが -*-courier-*-16* であるとか, oclock の widget は長針の色が赤で短針の色が柿色で, 縁の色が黒であるとか, mule のフォントは -*-marumoji-*--16-*-jisx0208.1983-0 であるとかです.

これらの属性をリソースといいます. そして, そのリソースを参照するために使用される名前をリソース名といいます. widget と同じようにリソースもそれぞれのリソースクラスに属しています.

さて, kterm はスクロールバーを使用していなければ, 普通 2 つの widget からできています. インスタンス名 (クラス名) です.

```
kterm(KTerm)
  |
  +--vt100(VT100)
```

kterm という widget の上を vt100 という widget が覆っています. 例えば次のようにして, kterm の vt100 の forground というリソースの値を青にできます.

```
% xrdb
kterm.vt100.foreground: Blue
^D
```

xrdb はリソースを X Server にロードするためのコマンドです. ~/.Xresources なども xrdb を使ってロードされます.

kterm では文字は vt100 widget の window に表示されますので, 文字が青くなります.

さて, XToolkit を使用していれば -name オプションでインスタンス名が変更できます.

kterm -name console を実行してみて下さい. 文字は青くありませんね. これを青くするためには,

```
console.vt100.foreground: Blue
```

としなければなりません.

ところが, 普通はインスタンス名にこだわらず, kterm の文字の色を一括して指定したいと思うでしょう. そういう時はクラス名を指定します.

```
KTerm.vt100.foreground: Blue
```

こうしておけば, kterm の文字の色はいつも青くなります. ただし, 次のように,

```
login.vt100.foreground: Red
```

としておいて, kterm -name login とすれば文字の色は赤となります. リソースを決定する時はより詳しく指定されている方が採用されるのです.

また, リソース名 foreground をクラス名 Foreground で書くこともできます.

さて, 基本的にはこれでいいのですが, リソース名, インスタンス名, クラス名はワイルドカードにより省略ができます. 例えば,

```
kterm.?.foreground: Red
```

とか,

```
kterm*foreground: Red
```

とかです. 意味は分かりますね. さて, この時どうなるのでしょう ? 実は kterm は vt100 端末のエミュレーション機能だけでなく, tektronix 端末のエミュレーション機能ももっています. tek4014 端末をエミュレートしている時は,

```
kterm(KTerm)
  |
  +--tek4014(Tek4014)
```

となっています. kterm.vt100.foreground では, vt100 の方の色しか指定されていませんが, kterm.?.foreground や kterm*foreground では当然ワイルドカードに tek4014 もマッチしてしまいますので, 線の色が赤になります.

そして, 省略が極端になると,

```
*Foreground: Green
```

などという指定もでてきます, これは, Foreground クラスに属するリソースの値を緑にするという指定です. 例えば, oclock の短針と長針の色はそれぞれ, oclock.clock.hour とoclock.clock.minute で指定されますが, どちらもクラスは Clock.Clock.Foreground なので緑色になってしまいます.

4 さて具体的には

次の xtsample.cc を利用して説明することにします. C++ プログラムですが, C++ の機能で利用しているのは, // によるコメントと, 構造体の前にいちいち struct を付ける必要がないこと, ローカル変数はブロックの先頭でなくても宣言できることだけですから, C しか知らない人でも大丈夫だと思います.

実際に利用してみるとリソースの利用の仕方などが良く分かりますので, 駒場のアカウントがある人は, ~g440604/c/xtsample/ の下の xtsample を実行してみて下さい.

とりあえす, 実行すると日の丸と右上にアメリカの国旗がでるはずです. リソースで変な設定をしている人は, 色が変かもしれません. 大きい旗の上では [space] で旗の模様が変わり, 小さい旗の上では j, a, f で大きい旗の種類を直接していできます. q か [ESC] か, Window Manager の close(fvwm), delete(twm) で正常終了します.

つぎにリソースを書いてみます.

```
% xrdb
xtsample*translations: #override \n\
        <Key>q:         change()resize()redraw(ALL)\n\
        <Key>space:     quit()
xtsample*foreground:    Orange
^D
```

どうでしょう ? さっきとは, [space] と, q の機能が入れ替わっていますね. 又, 日の丸の色がオレンジになっています. 同じことが,

```
% xtsample -fg Orange -xrm 'xtsample*translations:#override\n<Key>q:
                              change()resize()redraw(ALL)\n<Key>space:quit()'
```

でも指定できます.

xtsample では, widget tree は次のようになっています.

```
xtsample(XTsample)
  |
  +--flag(Composite)
       |
       +--miniFlag(Core)
```

Composite という widget class は子 widget を複数持てる widget の一番基本のものです. Core という widget class は widget の最も基本なものです. (全ての widget は Core から派生しています)

では, ソースを見ていきましょう.

xtsample.cc

```
001 //
002 // Xt Sample - GANA
003 // xtsample.cc
004 //
005
006
007 #include <X11/IntrinsicP.h>       // ふつうはいらない
008 #include <X11/Intrinsic.h>        // X toolkit insrinsics
009 #include <X11/StringDefs.h>       // リソース名の定義
010 #include <X11/Shell.h>            // shell widget
011 #include <X11/Composite.h>        // composite widget
012 #include <X11/Core.h>             // core widget
```

```
013 #include <X11/extensions/shape.h> // shape window
014 #include <stdlib.h>
015 #include <stdio.h>
016
017
018 #include "icon.xbm"
019 #include "mask.xbm"
```

この二つはアイコンを表示するためのデータです. このソースの後ろに載しておきます.

```
020
021
022 // ディフォルトリソース
023 static String default_resources[]=
024 {
025   "XTsample.Title:       Xt Sample",
026   "XTsample.Geometry:    11x7",
030   "XTsample*BorderWidth: 0",
031   NULL,
032 };
```

ここでこれらのリソースが最初から設定されているために, タイトルバーに “Xt Sample” と表示されたり, 日の丸が赤色だったりします.

```
033
034
035 // オプションの指定
036 static XrmOptionDescRec options[]=
037 {
038   //{ "-option", "resource", Xrmoption*,      (XPointer)value },
039 {"--help",       ".help",    XrmoptionNoArg,  (XPointer)"True"},
040 {"-help",        ".help",    XrmoptionNoArg,  (XPointer)"True"},
041 {"-h",           ".help",    XrmoptionNoArg,  (XPointer)"True"},
042 {"-f",           ".flag",    XrmoptionSepArg, (XPointer)NULL},
043 {"-flag",        ".flag",    XrmoptionSepArg, (XPointer)NULL},
044 {"-blue",        ".blue",    XrmoptionSepArg, (XPointer)NULL},
045 };
```

ここは, コマンドラインの自動解析に関するデータを指定しています. 例えば, `--help` が引数に指定されると, `xtsample.help` のリソースが `"True"` に設定されます. 又, `-flag 1` と指定されれば, `xtsample.flag` のリソースが `"1"` に設定されます. 3 番目の定数 `XrmOption...` によって, オプションの解析方法が変わってきます.

XrmOptionDescRec options[x]	コマンドライン	xtsample.option に設定されるリソース
{"-option", ".option", XrmoptionNoArg, "True"}	-option	"True"
{"-option", ".option", XrmoptionIsArg, NULL}	-option	"option"
{"-opt", ".option", XrmoptionStickyArg, NULL}	-option	"ion"
{"-option", ".option", XrmoptionSepArg, NULL}	-option this	"this"
{"-option", ".option", XrmoptionResArg, NULL}	-option "*a:b"	名称 a の全リソースに "b"
{"-option", ".option", XrmoptionSkipArg, NULL}	-option skip	この 2 つを無視
{"-option", ".option", XrmoptionSkipLine, NULL}	-option ignore	この先全てを無視

XrmOptionDescRec は以下のように定義されています.

```
typedef struct {
    char            *option;    /* Option abbreviation in argv        */
    char            *specifier; /* Resource specifier                 */
    XrmOptionKind   argKind;    /* Which style of option it is        */
    XPointer        value;      /* Value to provide if XrmoptionNoArg */
} XrmOptionDescRec, *XrmOptionDescList;
```

```
046
047
048 // shell widget のリソースの内容が読み込まれる構造体
049 static struct ShellAppData
050 {
051   Bool   help;
052   String foreground;
053   String background;
054   String blue;
055   int    flag;
056   int    base_size;
057   int    inc_size;
058 }shell_app_data;
059
060
061 // 前出の構造体のどのメンバにどのリソースを読み込むか
062 static XtResource shell_resources[]=
063 {
064 { "help",       "Help",       XtRBoolean, sizeof(Boolean),
        XtOffset(ShellAppData*, help),       XtRImmediate, (XtPointer)False},
065 { "foreground", "Foreground", XtRString,  sizeof(String),
        XtOffset(ShellAppData*, foreground), XtRString,    (XtPointer)"Red"},
066 { "background", "Background", XtRString,  sizeof(String),
        XtOffset(ShellAppData*, background), XtRString,  (XtPointer)"White"},
067 { "blue",       "Blue",       XtRString,  sizeof(String),
        XtOffset(ShellAppData*, blue),       XtRString,   (XtPointer)"Blue"},
```

```
068 { "flag",       "Flag",       XtRInt,     sizeof(int),
        XtOffset(ShellAppData*, flag),       XtRImmediate, (XtPointer)0},
069 { "baseSize",   "BaseSize",   XtRInt,     sizeof(int),
        XtOffset(ShellAppData*, base_size),  XtRImmediate, (XtPointer)200},
070 { "incSize",    "IncSize",    XtRInt,     sizeof(int),
        XtOffset(ShellAppData*, inc_size),   XtRImmediate, (XtPointer)32},
071 };
```

333 行で， ShellAppData 構造体にリソースを読み込みますが， その時どのリソースをどの変数に読み込むかを指定するものです． リソースのディフォルトの値は, default_resources で設定できるだけでなくここでも設定できます.

XtResource は以下のように定義されています.

```
typedef struct _XtResource {
    String    resource_name;   /* Resource name                              */
    String    resource_class;  /* Resource class                             */
    String    resource_type;   /* Representation type desired                */
    Cardinal  resource_size;   /* Size in bytes of representation            */
    Cardinal  resource_offset; /* Offset from base to put resource value     */
    String    default_type;    /* representation type of specified default */
    XtPointer default_addr;    /* Address of default resource                */
} XtResource, *XtResourceList;
```

```
072
073 static Widget shell;          // shell widget
074 static Window shell_w=0;      // shell widget の window ID
```

shell widget というのは, 自分の作るアプリケーションのウィンドウの中で最も親となるべき widget です. つまり, window になった時にタイトルバーがついたりする widget です. インスタンス名は xtsample でクラス名は Shell ではなくて, XTsample となります.

```
075
076 static Widget composite;      // composite widget
077 static Window composite_w=0;  // composite widget の window ID
078 static GC     composite_gc=0; // composite widget の GC
079
080 static Widget core;           // core widget
081 static Window core_w=0;       // core widget の window ID
082 static GC     core_gc=0;      // core widget の GC
083
084 static Pixmap shape_pxm=0;    // shape の pixmap の ID
085 static GC     shape_gc=0;     // shape の GC
086 static int    can_use_shape=True; // shape extension は使用可能？
```

この shape_pxm で, ウィンドウにマスクをかけて, 矩形でないウィンドウを作ります. 又, shape extension 機能が使えない X Server も存在します.

```
087
088 static XtAppContext acr;       // なんでしょ？これ。
089 static Display      *d=NULL;  // display 構造体
090
091 static Pixmap icon, mask;      // icon の絵の白黒 pixmap ID
092 static XColor fore, back, blue; // それぞれの色のパレット番号
093 static Atom   atom_wm_protocols, atom_wm_delete_window;
```

この Atom は, Window Manager が送ってくるメッセージのうち, WM_DELETE_WINDOW の Atom です. (ある X Server の中では, 色々な定数や文字列が色々な用途に使用されていますが, それらを一意に識別するために, それらに唯一の数を与えています. Atom とは, その数のことです.)

```
094
095
096 static void get_window_size
          (const Window w, unsigned int *width, unsigned int *height)
097 {
098   Window root;
099   int x, y;
100   unsigned int border, depth;
101   XGetGeometry(d, w, &root, &x, &y, width, height, &border, &depth);
102 }
103
104
105 static void japan(Window w, GC gc)
106 {
107   unsigned int width, height;
108   get_window_size(w, &width, &height);
109   int r=((width<height)?width:height)*3/8;
110   XSetForeground(d, gc, back.pixel);
111   XFillRectangle(d, w, gc, 0, 0, width, height);
112   XSetForeground(d, gc, fore.pixel);
113   XFillArc(d, w, gc, width/2-r, height/2-r, r*2, r*2, 0*64, 360*64);
114 }
115
116
117 static void america(Window w, GC gc)
118 {
119   unsigned int width, height;
120   get_window_size(w, &width, &height);
121   XSetForeground(d, gc, back.pixel);
122   XFillRectangle(d, w, gc, 0, 0, width, height);
123   XSetForeground(d, gc, fore.pixel);
124   int h=height/13;
125   for (int i=0; i<height; i+=h+h)
126     XFillRectangle(d, w, gc, 0, i, width, h);
127   XSetForeground(d, gc, blue.pixel);
128   XFillRectangle(d, w, gc, 0, 0, width/2, h*7);
129 }
130
131
```

```
132 static void france(Window w, GC gc)
133 {
134   unsigned int width, height;
135   get_window_size(w, &width, &height);
136   XSetForeground(d, gc, fore.pixel);
137   XFillRectangle(d, w, gc, 0, 0, width, height);
138   width/=3;
139   XSetForeground(d, gc, blue.pixel);
140   XFillRectangle(d, w, gc, 0, 0, width, height);
141   XSetForeground(d, gc, back.pixel);
142   XFillRectangle(d, w, gc, width, 0, width, height);
143 }
144
145
146 static void composite_redraw(void)
147 {
148   switch (shell_app_data.flag)
149   {
150   case 0:
151     japan  (composite_w, composite_gc); break;
152   case 1:
153     america(composite_w, composite_gc); break;
154   case 2:
155     france (composite_w, composite_gc); break;
156   }
157 }
158
159
160 static void shell_resize(void)
161 {
162   unsigned int width, height;
163   get_window_size(shell_w, &width, &height);
164   if (can_use_shape)
165   {
166     if (shape_gc) XFreeGC(d, shape_gc);
167     if (shape_pxm) XFreePixmap(d, shape_pxm);
168     shape_gc=0;
169     shape_pxm=0;
170   }
171   switch (shell_app_data.flag)
172   {
173   case 0:
174   case 1:
175     if (can_use_shape)
176     {
177       // pixmap に 0 を指定しても、shape が解除される訳ではないらしい
178       XShapeCombineMask(d, shell_w, ShapeBounding, 0, 0, 0, ShapeSet);
179       XShapeCombineMask(d, shell_w, ShapeClip,     0, 0, 0, ShapeSet);
180     }
181     break;
182   case 2:
183     if (can_use_shape)
184     {
185       // shape extension を使用して、window をくりぬきます
```

```
186       // どの部分をくりぬくかをきめる白黒 pixmap を作成
187       shape_pxm=XCreatePixmap(d, shell_w, width, height, 1);
188       shape_gc=XCreateGC(d, shape_pxm, 0, 0);
189       // 色 1 で描画した部分は window となって残る
190       // 色 0 で描画した部分がくりぬかれる
191       int h=height/16;
192       XSetForeground(d, shape_gc, 1);
193       XFillRectangle(d, shape_pxm, shape_gc, 0, 0, width, height);
194       XSetForeground(d, shape_gc, 0);
195 #define oval(X,Y,W,H) \
196       XFillArc(d, shape_pxm, shape_gc, width/2+h*(X), h*(Y),
                                              h*(W), h*(H), 0*64, 360*64);
197       oval(-5,  1, 10, 10);
198       oval(-4,  4,  8,  9);
199       XSetForeground(d, shape_gc, 1);
200       oval(-3,  4, 2, 4);
201       oval( 1,  4, 2, 4);
202       oval(-1,  8, 1, 1);
203       oval( 0,  8, 1, 1);
204       oval(-3, 10, 6, 1);
205
206       XSetForeground(d, shape_gc, 0);
207       XShapeCombineMask(d, shell_w, ShapeBounding, 0, 0,
                                                shape_pxm, ShapeSet);
208       XShapeCombineMask(d, shell_w, ShapeClip,     0, 0,
                                                shape_pxm, ShapeSet);
209     }
210     break;
211   }
212   XtResizeWidget(core, width/10, height/10, 0);
213   XtMoveWidget(core, width-width/10, 0);
214 }
```

この辺りが shape extension による window のくりぬき処理です. shell_pxm に絵を書いて, それをマスクとして window (shell_w) にセットしています. この辺は oclock のソースからの見よう見まねなので何が行なわれているのかいまいちよく分かっていません. X の man page も, “This manual pages needs a lot more work.” と言ってほとんど何も書いてありません.

```
void XShapeCombineMask (
    Display*, /* display   */
    Window,   /* dest      */
    int,      /* dest_kind */
    int,      /* x_off     */
    int,      /* y_off     */
    Pixmap,   /* src       */
    int       /* op        */
);
```

```
215
216
217 static void core_redraw(void)
218 {
219   switch (shell_app_data.flag)
220   {
221   case 0:
222     america(core_w, core_gc); break;
223   case 1:
224     france (core_w, core_gc); break;
225   case 2:
226     japan  (core_w, core_gc); break;
227   }
228 }
229
230
231 static void change_flag
          (Widget widget, XEvent *e, String *params, Cardinal *num_params)
232 {
233   if (*num_params>0) shell_app_data.flag=atoi(params[0])%3;
234   else               shell_app_data.flag=(shell_app_data.flag+1)%3;
235   fprintf(stderr, "flag changed to %d\n", shell_app_data.flag);
236 }
237
238
239 static void quit
          (Widget widget, XEvent *e, String *params, Cardinal *num_params)
240 {
241   fprintf(stderr, "quit\n");
242   exit(0);
243 }
244
245
246 static void redraw
          (Widget widget, XEvent *e, String *params, Cardinal *num_params)
247 {
248   fprintf(stderr, "redraw\n");
249   if (*num_params>0)
250   {
251     composite_redraw();
252     core_redraw();
253   }
254   else if (widget==composite) composite_redraw();
255   else if (widget==core) core_redraw();
256   XFlush(d);
257 }
258
259
260 static void resize
          (Widget widget, XEvent *e, String *params, Cardinal *num_params)
261 {
262   fprintf(stderr, "resize\n");
263   shell_resize();
264 }
265
266
```

```
267 static void close_window
          (Widget widget, XEvent *e, String *params, Cardinal *num_params)
268      // window manager からの delete window メッセージ
269      // (fvwm なら close 、twm なら delete) を受けとった
270 {
271   if (e->type==ClientMessage &&
272       e->xclient.message_type==atom_wm_protocols &&
273       e->xclient.data.l[0]==atom_wm_delete_window)
274   {
275     fprintf(stderr, "close\n");
276     exit(0);
277   }
278 }
```

ここで, 93 行目の Atom を使用しています.

```
279
280
281 static XtActionsRec actions[]=
282 {
283 {"quit",   quit},
284 {"change", change_flag},
285 {"redraw", redraw},
286 {"resize", resize},
287 {"close",  close_window},
288 };
```

ここで, 実際の関数と, リソースで設定される関数名を対応させています.

```
typedef struct _XtActionsRec{
    String  string;
    XtActionProc proc;
} XtActionsRec;


typedef void (*XtActionProc)(
    Widget,   /* widget     */
    XEvent*,  /* event      */
    String*,  /* params     */
    Cardinal* /* num_params */
);
```

```
289
290
291 static String shell_translation=
292 "<ClientMessage>:   close()\n"
293 "<ConfigureNotify>: resize()redraw(ALL)\n";
294
295
296 static String composite_translation=
297 "<Key>space:        change()resize()redraw(ALL)\n"
```

```
298 "<Key>q:            quit()\n"
299 "<Key>Escape:       quit()\n"
300 "<Expose>:          redraw()\n";
301
302
303 static String core_translation=
304 "<Key>j:            change(0)resize()redraw(ALL)\n"
305 "<Key>a:            change(1)resize()redraw(ALL)\n"
306 "<Key>f:            change(2)resize()redraw(ALL)\n"
307 "<Key>q:            quit()\n"
308 "<Key>Escape:       quit()\n"
309 "<Expose>:          redraw()\n";
```

イベントが起こった時，どの関数がどんな引数でどういう順番に呼ばれるかを指定します.

translation に指定するもの	X Event	どういうイベントか
BtnDown, Btn1Down, Btn2Down ···	ButtonPress	マウスのボタンが押された
BtnUp, Btn1Up, Btn2Up ···	ButtonRelease	マウスのボタンが離された
Motion, PtrMoved, MouseMoved	MotionNotify	マウスが移動した
Enter, EnterWindow	EnterNotify	マウスがウィンドウ内に入った
Leave, LeaveWindow	LeaveNotify	マウスがウィンドウから出た
Key, KeyDown	KeyPress	キーが押された
KeyUp	KeyRelease	キーが離された
FocusIn	FocusIn	フォーカスを得た
FocusOut	FocusOut	フォーカスを失った
KeyMap	KeymapNotify	全キーの状態を検知した
Mapping	MappingNotify	キーボードのマップ状態が変わった
Expose	Expose	最描画要求
GrExp	GraohicExpose	
NoExp	NoExpose	
Visible	VisibilityNotify	可視/不可視状態の変化
Clrmap	ColormapNotify	
Message	ClientMessage	WM からのメッセージなど
Prop	PropertyNotify	
SelClr	SelectionClear	
Select	SelectionNotify	
Circ	CirculateNotify	ウィンドウの重なり方の変化
Configure	ConfigureNotify	リサイズされた，など
Create	CreateNotify	ウィンドウが生成された
Destroy	DestroyNotify	ウィンドウが消滅した
Map	MapNotify	ウィンドウがマップされた
Unmap	UnmapNotify	ウィンドウがアンマップされた
Grav	GravityNotify	
Reparent	ReparentNotify	親ウィンドウの変更
CircReq	CirculateRequest	
ConfigureReq	ConfigureRequest	
MapReq	MapRequest	マップ要求
ResizeReq	ResizeRequest	リサイズ要求

表記例	意味
<Btn1Down>	ボタン 1 を押した
Shift<Btn1Down>	シフトを押しながらボタン 1 を押した
Ctrl<Btn1Up>	コントロールを押しながらボタン 1 を離した
<Btn1Down>(2)	ダブルクリック
<Btn1Down>(2+)	ダブルクリック以上
<Key>a	a キーを押した (大文字・小文字関係なし)
"Try"	Try と入力した
!Ctrl<Key>A	コントロールのみを押しながら a キーを押した
Button1<Key>a	ボタン 1 を押しながら a キーを押した
Shift<Key>a	シフトを押しながら a キーを押した
	Ctrl, Shift, Lock, Meta, Alt, Mod1, Button1 などが使える

```
310
311
312 int main(int argc, char *argv[])
313 {
314   // widgets
315   shell=XtVaAppInitialize
316     (&acr, "XTsample",
317      options, XtNumber(options),
318      &argc, argv,
319      default_resources, NULL);
```

まず, shell widget を作成します. クラス名を XTsample とし, インスタンス名はコマンドラインから自動的にせっていされます (大抵, xtsample となりますが, -name オプションなどが指定されれば別です). argc, argv を指定することによりコマンドラインの解析が行なわれ, 解析されたコマンドライン引数は argc, argv からとりのぞかれます. 又, X Server との接続を確立し, リソースを読み込みます. ただし, この時点では shell widget の window は作成されていません.

```
Widget XtVaAppInitialize(
    XtAppContext*,     /* app_context_return */
    _Xconst _XtString, /* application_class  */
    XrmOptionDescList, /* options            */
    Cardinal,          /* num_options        */
    int*,              /* argc_in_out        */
    String*,           /* argv_in_out        */
    String*,           /* fallback_resources */
    ...
);
```

```
320   composite=XtVaCreateManagedWidget
321     ("flag", compositeWidgetClass,
322      shell,
323      NULL);
324   core=XtVaCreateManagedWidget
325     ("miniFlag", coreWidgetClass,
326      composite,
327      XtNwidth,  40,
328      XtNheight, 30,
329      NULL);
```

flag と miniFlag を作成します. インスタンス名は, miniFlag のように, 先頭を子文字にすることが推奨されてます.

```
Widget XtVaCreateManagedWidget(
    _Xconst _XtString,  /* name         */
    WidgetClass,        /* widget_class */
    Widget,             /* parent       */
    ...
);
```

```
330
331   XtAppAddActions(acr, actions, XtNumber(actions));
```

ここで, 実際の関数と, リソースで設定される関数名の対応を登録しています. (281〜288 行目参照)

```
void XtAppAddActions(
    XtAppContext, /* app_context */
    XtActionList, /* actions     */
    Cardinal      /* num_actions */
);
```

```
332   d=XtDisplay(shell);
333   XtGetApplicationResources
334     (shell,
335      XtPointer(&shell_app_data),
336      shell_resources, XtNumber(shell_resources),
337      NULL, 0);
```

ここで, 61〜71 行で設定した情報に基づいてリソースの値を app_data 構造体に代入しています.

```
void XtGetApplicationResources(
    Widget,         /* widget        */
    XtPointer,      /* base          */
    XtResourceList, /* resources     */
    Cardinal,       /* num_resources */
    ArgList,        /* args          */
    Cardinal        /* num_args      */
);
```

```
338   if (shell_app_data.help || argc>1)
339   {
340     fprintf(stderr,
341             "usage: xtsample [options ...]\n"
342             "       -h, -help, -flag <n>, -blue <color>, ...\n");
343     return 1;
344   }
345
346   // translations
347   XtAugmentTranslations(shell,
                            XtParseTranslationTable(shell_translation));
348   XtAugmentTranslations(composite,
                            XtParseTranslationTable(composite_translation));
349   XtAugmentTranslations(core,
                            XtParseTranslationTable(core_translation));
```

291〜309 行目で設定したディフォルトの translation を登録します. ただし, 同じ translation が既に設定されていたら, それは登録しません.

```
void XtAugmentTranslations(
    Widget,        /*  widget      */
    XtTranslations /* translations */
);
```

```
350
351   // shape
352   int shape_event_base, shape_error_datas;
353   can_use_shape=XShapeQueryExtension(d, &shape_event_base,
                                                    &shape_error_datas);
```

shape extension が使用可能かどうかの問い合わせをします.

```
Bool XShapeQueryExtension (
    Display*, /* display    */
    int*,     /* event_base */
    int*      /* error_base */
);
```

```
354
355   // icon
356   icon=XCreateBitmapFromData(d, RootWindow(d, 0),
                                 (char *)icon_bits, icon_width, icon_height);
357   mask=XCreateBitmapFromData(d, RootWindow(d, 0),
                                 (char *)mask_bits, mask_width, mask_height);
358   XtVaSetValues(shell, XtNiconPixmap, icon, XtNiconMask, mask, NULL);
```

shell widget のリソースとしてアイコンを登録します. このリソースはWindow Manager によってアイコンとして利用されます.

```
void XtVaSetValues(
    Widget, /* widget */
    ...
);
```

```
359
360   // color
361   XColor dummy;
362   XAllocNamedColor(d, DefaultColormap(d, 0),
                                 shell_app_data.foreground, &fore, &dummy);
363   XAllocNamedColor(d, DefaultColormap(d, 0),
                                 shell_app_data.background, &back, &dummy);
364   XAllocNamedColor(d, DefaultColormap(d, 0),
                                 shell_app_data.blue,       &blue, &dummy);
```

リソースで指定された色の名前から, その色のパレット番号と, rgb の値を取得します.

```
365
366   // resources
367   XtVaSetValues
368     (shell,
369      // XtNwidth,     0,
370      // XtNheight,    0,
371      // XtNmaxWidth,  0,
372      // XtNmaxHeight, 0,
373      // XtNminWidth,  0,
374      // XtNminHeight, 0,
375      XtNbaseWidth,    shell_app_data.base_size,
376      XtNbaseHeight,   shell_app_data.base_size,
377      XtNwidthInc,     shell_app_data.inc_size,
378      XtNheightInc,    shell_app_data.inc_size,
379      XtNinput,        True,
380      NULL);
```

shell widget にさらに大きさなどのリソースを設定します. まず, base サイズは, これ以上ちじまない大きさで, inc サイズ単位で大きくなることができます. kterm のようなサイズの変更が可能になると考えて下さい.

コメントアウトされている, max と min を指定すればウィンドウの最大と最小の大きさが指定できます. base, inc との併用はできないようです.

XtNwidth と XtNheight はウィンドウが生成された時の大きさですが, これは, 26 行目で既に設定してあるのでここでは設定しません.

XtNinput はマウスがウィンドウ内に入った時フォーカスを渡すように Window Manager に指示しています. こうしておかないと, Window Manager はフォーカスを渡してくれません.

```
381   XtRealizeWidget(shell);
```

ここでやっと 3 つの widget の window が実際に生成されます.

```
void XtRealizeWidget(
    Widget /* widget */
);
```

```
382
383   shell_w     =XtWindow(shell);
384   composite_w =XtWindow(composite);
385   composite_gc=XCreateGC(d, composite_w, 0, 0);
386   core_w      =XtWindow(core);
387   core_gc     =XCreateGC(d, core_w, 0, 0);
388
389   // close
390   atom_wm_protocols    =XInternAtom(d, "WM_PROTOCOLS",     False);
391   atom_wm_delete_window=XInternAtom(d, "WM_DELETE_WINDOW", False);
392   XSetWMProtocols(d, shell_w, &atom_wm_delete_window, 1);
```

Window Manager に, WM_DELETE_WINDOW メッセージを送るように指示しています. (93, 267〜278 行目参照)

```
393
394   XtAppMainLoop(acr);
```

イベント待ちループに入ります. この関数は制御を返しません. だから, 395 行目に到達することは決してないのです.

```
395 }
```

次の icon.xbm と mask.xbm は bitmap で生成しました. xv などでも生成可能です.

icon.xbm

```
001 #define icon_width 49
002 #define icon_height 25
003 static unsigned char icon_bits[] = {
004    0x00, 0x7c, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x00, 0x88, 0x00, 0x0a, 0x0f,
005    0x00, 0x00, 0x00, 0x10, 0x01, 0x85, 0x08, 0x00, 0x00, 0x00, 0x20, 0x82,
006    0xe2, 0x38, 0x00, 0x00, 0x00, 0x40, 0x44, 0x21, 0x20, 0x00, 0x00, 0x00,
```

```
007    0x80, 0xa8, 0xe0, 0x38, 0x00, 0x00, 0x00, 0x00, 0x55, 0x80, 0x08, 0x00,
008    0x00, 0x00, 0x80, 0x8a, 0x80, 0x08, 0x00, 0x00, 0x00, 0x40, 0x11, 0x81,
009    0x08, 0x00, 0x00, 0x00, 0xa0, 0x20, 0x82, 0x08, 0x00, 0x00, 0x00, 0x50,
010    0x40, 0x84, 0x30, 0x00, 0x00, 0x00, 0x28, 0x80, 0x08, 0x21, 0x00, 0x00,
011    0x00, 0x1c, 0x00, 0x1f, 0x3e, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
012    0x00, 0x00, 0x7e, 0x3e, 0xee, 0xfc, 0x3c, 0xfc, 0x01, 0x41, 0x41, 0x11,
013    0x05, 0x25, 0x04, 0x01, 0x79, 0x49, 0x11, 0x25, 0x25, 0xe4, 0x01, 0x09,
014    0x49, 0x11, 0x25, 0x25, 0x24, 0x00, 0x39, 0x49, 0x11, 0x25, 0x25, 0xe4,
015    0x00, 0x41, 0x41, 0x01, 0x05, 0x25, 0x84, 0x00, 0x4e, 0x49, 0x29, 0xe5,
016    0x24, 0xe4, 0x00, 0x48, 0x49, 0x29, 0x25, 0x24, 0x24, 0x00, 0x4f, 0x49,
017    0x29, 0x25, 0xe4, 0xe5, 0x01, 0x41, 0x49, 0x29, 0x25, 0x04, 0x05, 0x01,
018    0x3f, 0x7f, 0xff, 0x3d, 0xfc, 0xfd, 0x01};
```

mask.xbm

```
001 #define mask_width 49
002 #define mask_height 25
003 static unsigned char mask_bits[] = {
004    0x00, 0x7c, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x00, 0xf8, 0x00, 0x0e, 0x0f,
005    0x00, 0x00, 0x00, 0xf0, 0x01, 0x87, 0x0f, 0x00, 0x00, 0x00, 0xe0, 0x83,
006    0xe3, 0x3f, 0x00, 0x00, 0x00, 0xc0, 0xc7, 0xe1, 0x3f, 0x00, 0x00, 0x00,
007    0x80, 0xef, 0xe0, 0x3f, 0x00, 0x00, 0x00, 0x00, 0x77, 0x80, 0x0f, 0x00,
008    0x00, 0x00, 0x80, 0xfb, 0x80, 0x0f, 0x00, 0x00, 0x00, 0xc0, 0xf1, 0x81,
009    0x0f, 0x00, 0x00, 0x00, 0xe0, 0xe0, 0x83, 0x0f, 0x00, 0x00, 0x00, 0x70,
010    0xc0, 0x87, 0x3f, 0x00, 0x00, 0x00, 0x38, 0x80, 0x0f, 0x3f, 0x00, 0x00,
011    0x00, 0x1c, 0x00, 0x1f, 0x3e, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
012    0x00, 0x00, 0x7e, 0x3e, 0xee, 0xfc, 0x3c, 0xfc, 0x01, 0x7f, 0x7f, 0xff,
013    0xfd, 0x3d, 0xfc, 0x01, 0x7f, 0x7f, 0xff, 0xfd, 0x3d, 0xfc, 0x01, 0x0f,
014    0x7f, 0xff, 0xfd, 0x3d, 0x3c, 0x00, 0x3f, 0x7f, 0xff, 0xfd, 0x3d, 0xfc,
015    0x00, 0x7f, 0x7f, 0xff, 0xfd, 0x3d, 0xfc, 0x00, 0x7e, 0x7f, 0xff, 0xfd,
016    0x3c, 0xfc, 0x00, 0x78, 0x7f, 0xff, 0x3d, 0x3c, 0x3c, 0x00, 0x7f, 0x7f,
017    0xff, 0x3d, 0xfc, 0xfd, 0x01, 0x7f, 0x7f, 0xff, 0x3d, 0xfc, 0xfd, 0x01,
018    0x3f, 0x7f, 0xff, 0x3d, 0xfc, 0xfd, 0x01};
```

さて, プログラムを作ったならば, 次はコンパイルです. しかし, X のライブラリは ld (リンカ) のサーチパスから外れていたり, socket のライブラリを要求したりと, 機種によって様々でいろいろ面倒です. そこで, うまくコンパイルできるように Imakefile というものを使います.

Imakefile

```
001 LOCAL_LIBRARIES = $(XLIB) $(XTOOLLIB)
002 DEPLIBS = $(DEPXLIB) $(DEPXTOOLLIB)
003 OBJS = xtsample.o
004 SRCS = xtsample.cc
005
006 ComplexProgramTarget(xtsample)
```

これを作ってから

```
% xmkmf
% make depend
% make
```

とします (このプログラムの場合は depend の必要はありませんが). すると, xmkmf により, 適当なライブラリがリンクされるような Makefile が生成され, make によりコンパイル, リンクされます.

ふぅ. 結構しんどいですね.

5 XToolkit を使いたくなったでしょう ?

なんかすごく面倒そうに見えますが, Xlib だけで作った時よりはソフトウェアの使い勝手がかなり良くなると思います. Xlib しか使ったことのない人はぜひ XToolkit を使ってみましょう.

でわ〜

編集後記

ううむ、原稿集めはめんどくさいなぁ。自分としては先手を打ったつもりだったのだが。やはり出る日が決っている部報はみんなぎりぎりまで大丈夫だと思うらしい。うぅむ。

全体のデザインが思い付かなくてちょもらんまさんのデザインを踏襲してしまった。次号までになんか考えます。

何はともあれ1年間よろしく。 (Aleph-NULL)

理論科学グループ　部報 196 号

1995 年 12 月 16 日　初版　第 1 刷発行

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196号

— クリスマスコンパ号 —

1995**年**12**月**16**日**

THEORETICAL SCIENCE GROUP