

愛知工業大学システム工学研究会

**会誌「\*ISM 02」**

**2010年度春季 ＆ 2010年度卒業生記念号**

-◆ 目次 ◆-

○ \*ISM 02に寄せて

○ \*ISMとは何か。

○ シス研の昨年の活動状況

○ H8/3069Fマイコン入門のススメ

○ 特集記事「車に詳しくなろう」

○ リバースエンジニアリングの実践

そのに～日本語化～

○ PDAを用いたGPS信号の取得

○ Open GL で図形を描こう

○ OpenLDAP による Unix , Linux

総合認証システム　第一回

○ 工科展作業班の報告

○ 役員からの一言

○ 編集後記

○ シス研への連絡方法

\*ISM 02に寄せて

10年度卒業　立松裕將（株式会社エヌ・ティ・ティ ネオメイト）

システム工学研究会会誌\*ISM第2号の発行、おめでとうございます。08年度会長

の立松です。昨年は、Make Tokyo Meetingへの作品出展や、他大学サークルとの

技術交流も始まり、内部の活動から外部の活動へと本格的に昇華する、たいへん

良い機会を得ることができました。外部の活動というのはごまかしの効かないも

のです。ごまかしの効かない環境で得た実力と経験は、必ず後輩諸君の今後の就

職活動、研究活動の支えとなるでしょう。特に就職難の時代ですから、自らの選

択の枠を増やすためにも、「苦労は買ってでもしろ」の勢いで頑張って欲しいと

思います。

さて、本年度は部室の移転、新たな活動への参加など、システム工学研究会にと

って変遷の年となりそうです。どのような形になるのかはまだわかりませんが、

それぞれが精一杯の力を発揮し、システム工学研究会の明日を重ねて行ってくだ

さい。以上で、私からの会誌発行の挨拶文とさせて頂きます。

07年度入学 元会長 奥村洋介(経営情報科学部情報科学科)

09年度会長の奥村洋介です。

会誌「\*ISM」の二度目の発行あたり、ご挨拶させて頂きます。

まず、制作に携わった人たち、ありがとう。

本当は私の代で出したかったんだけれど、結局できずに申し訳ない。

お疲れ様でした。

今回私は記事をかいていませんが、皆の記事を楽しませて頂きます。

さて、あまり堅苦しいと読まれない気がするので、ここからは口調を変えていき

ましょうか。

これを書いている現在（2010年3月）、シス研は転換期とも言える時期だ。

そんな時期に就活で手一杯なこの状況が歯がゆくて悔しい。

けれど後任の瀧本やそのブレイン（？）である佐原をはじめとする後輩たちが頼

りになるので、あまり心配はしていません。

就活で烏合の衆と化している私たちの分も補なって、きっと最適な道を見つけ出

すはず。

シス研本来の在り方をもう一度考えてみるのもいいかもしれないね。

私が今までシス研で活動する上で思っていたのは

みんな謙虚だなぁ

ということ。

新しい事に触れたり挑戦するのは難しいけれど、その環境は用意されているから、

やりたいことやっていいよ！

もっとはっちゃけていいよ！

そういう雰囲気作りがいまいちできなかったのが心残っちゃ心残り。

それは私たちの責任だね。

なんだか卒業生のセリフみたいになってるな。

シス研での活動は私の大学生活の中で、大部分のウェイトを占めています。

それなしでは私の大学生活、いや人生は語れません。

辛いことや衝突もあったけれど、それも含めて入ってよかったと思っています。

会員全員が、後からそんな風に思えるように、みんなでシス研を作っていきま

しょう。

08年度入学 会長 瀧本将大(工学部電子情報科)

平成22年度会長の瀧本将大です。まず、今回の会誌発行に当たり記事を寄稿して下さった方々とこの会誌を手に取って頂いた方に感謝いたします。

私が昨年度、会長という役職を拝命して以来私の生まれる前から続いてきたサークルの

伝統や役割の事を考えると、私はこの職をうまくこなせているだろうかと不安に思うこともありました。しかし、今回沢山の方から戴いた記事に目を通しますと、会員一人ひとりがシス研の事を考えて行動していることが伝わりました。この調子ならば部室を移転したとしても、シス研のこの雰囲気のままでやっていくことが出来るだろうと思います。

本年度はシス研にとって大変試練な年になるだろうと思います。けれども部室が代わってしまう時もただヤドカリの様に居心地の良い貝殻を移動するだけでなく、自分自身を守ってくれた殻を脱皮する蟹の様に成長して、次代を任せられるシス研になっていきましょう。

\*ISMとは何か。(シス研とは何か。)

　\*ISMとはシステム工学研究会、略してシス研の会誌でありシス研の活動目的である「本会は、電子情報通信に関わる広範な技術に興味を持ち、各々の鍛錬を補助し、情報処理技術者試験の受験を目的とする。また成果を工科展に顕す」の実践を紹介する場所である。

ではそもそもシス研とは何か、という話となる。

　シス研とは、愛知工業大学の技術系サークルである。その活動範囲はハードとソフトを問わず、技術系としかいいようが無い。会員も個性的な人たちの集まりで、ギークな人たちやデスマーチが大好きな人たち、酒を飲むと人が変わってしまう人やら色々。

　そんな人達が集まって寝泊まったり、コーディングしたり、合宿に行ったり、飲みに行ったりするのがシス研である。

シス研の昨年の活動状況

5月 新入生歓迎会

バーベキュー

Make: Tokyo Meeting 03見学 (http://jp.makezine.com/)

6月

7月 鍋パーティー

8月 夏季休業

工科展作業

9月 合宿

工科展作業

10 大学祭

工科展

11月 Make: Tokyo Meeting 04参加

小旅行（下呂温泉）

12月 クリスマスパーティ

1月

2月 （遅い）新年会

第１回ネットワーク講習会（システム工学研究会主催、他大学の学生も参加）

3月 追い出し会

H8/3069Fマイコン入門のススメ

１EV　hekki

１．はじめに

皆さんはマイコンというものをご存知でしょうか？

マイコンとはマイクロコンピューターの略で、その名の通り小さいコンピュータな訳です。

そのマイコンを入学当初「C言語？マイコン？なにそれおいしーの？」状態だった僕が弄っていこうということです。

そこは違うだろ・もっとこうしたほうがいいと思う点も多々あるかもしれませんが、生暖かい目で記事を読んで頂けると幸いです。

２．使用機材

さて、使用機材ですが秋月電子で販売されているH8/3069Fマイコン（以後H8マイコン）を搭載したネット対応マイコンLANボードというものを使用します。このボードはH8マイコン本体とネットワークコントローラを実装しているので、ネットワークプログラミングも手軽に挑戦ができるそうです（今回はやりませんが…）。

あと74HC595APというシフトレジスタも使用します。

H８マイコンの入門・開発に関しては以下のサイトを参考にしました。

やまねこのマイコン実験室（http://wiki.livedoor.jp/yamamaya\_com/）

３．目標

７セグLEDの制御！

最初の目標は１～９の等間隔で順番に表示だったのですが、それだけじゃおもしろくねーぞと言われたので表示する数字・感覚をランダムにするというところまでを目標にしました。

４．実際の手順

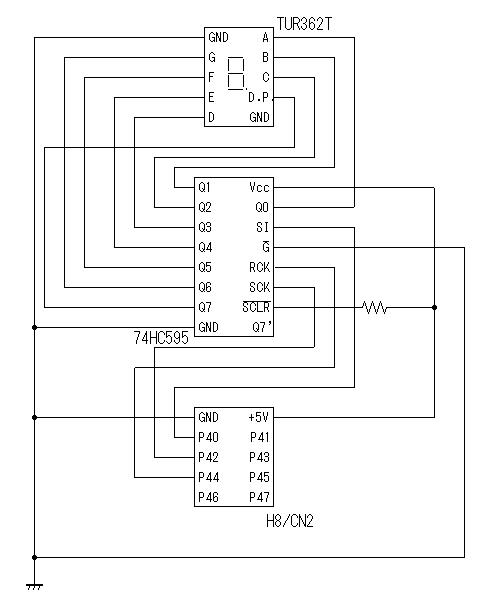
ここは僕が実際に作成をした手順を追って、苦労した点などを書いていきます。

最初にシフトレジスタそのものを理解する必要があったのですが、これが大変でした。しかし74HC595APはラッチが内蔵されているシフトレジスタなので、使い勝手が良く重宝すると思います。ここはグーグル先生と先輩達に泣きついてなんとか理解。

その後7セグLEDのアノードコモン・カソードコモンにちょっとビビりましたが、これはそんなに難しいことは無いです。

この時点でまずは先に回路の設計を進めて行きます。

回路図は下のほうに載せておきましたのでどうぞ。回路図を書くのにも手間取ってしまったので見にくいかもしれませんが、ご愛嬌ということで…ｗ



回路図の設計が終わったらいよいよソースコードの作成に入ります。

アセンブラに関してはチンプンカンプンだったので今回はC言語での開発を進めました。

ここで問題になってくるのがH8内臓のフラッシュROMの書き換え可能回数は１００回程度という問題です。

そこで、みついわゆきお氏が作成したMES（Micro Embeded System）というマイコン用のOSを使用し、ボード上のRAMに実行ファイルをダウンロードする、という形を取ります。

これで万事解決、あとはどんどん開発を進めていくだけです。

ここはサクっと書きましたが、一番苦労したところだと思いますｗ

ターミナルソフトの設定でコケたり、CUIを使ったことの無い僕にとってRAMに実行ファイルをダウンロードするのも一苦労でした（最終的に先輩に泣きつきました…）。

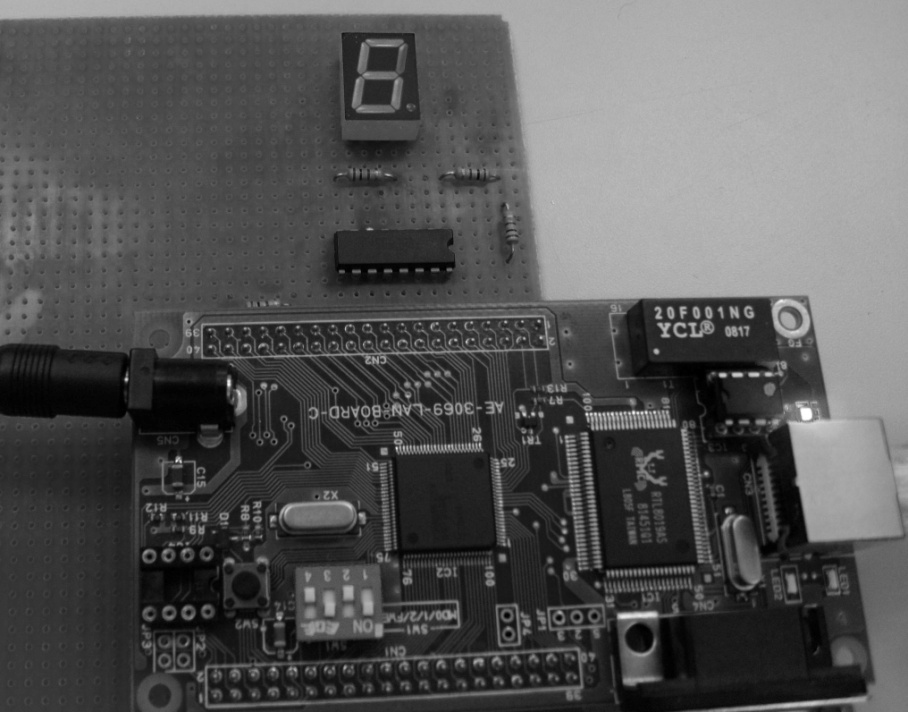
ただ、先程紹介したやまねこのマイコン実験室やMESに関しても本家サイトをよーく読めば書いてあることなので迷ったらとにかく読んでみることが大切だと思います。

あとはリファレンス・マニュアルを有効活用することの大切さも感じました。

ソースコードに関して一番苦労したのは乱数の生成に関してです。任意の数字を等間隔で表示、というところまでは結構すんなりできたのですが。。。

Windowsアプリケーションと違ってtime関数は使えないので乱数の種がサクッと出来ません。調べてみたところ、NTPサーバを使ってみたり、H8マイコンにはA/D変換が搭載されているのでノイズを拾ってみたりと色々あったのですがそれは今後の課題ということで（ぇ

今回はxorshiftという擬似乱数生成アルゴリズムを使ってみました。あくまで擬似乱数なので再帰性がありますが、サクっと実験するにはこれで十分かなと思いますｗ

５．まとめ

途中嫌になって、おうちに帰って寝たいお…と思うこともありましたが、完成したときは本当に嬉しいものです。これはやめられないな、と感じた瞬間でした。

正直いって先輩方の助け・同級生のおもしろくねーぞ改良しろ！の声が無ければ、完成はしなかったと思います。この場を借りてお礼をしたいと思います。

最後に動作風景をどうぞ！

車に詳しくなろう

工学部電気学科情報通信専攻2年 倉地和哉

最初に一言

○「シス研なのになぜ車？」と思って読む人もいるではないでしょうか…

簡単に言うと愛工大は自家用車とバイクでの通学が許されています。シス研会員も多くの人が車やバイクを所有しています。そして、サークル活動の行事や買い出し等に使用しています。そこで、今回記事にさせてもらいました。

本題です

○車には色々な形や種類、メーカーがあり、わかりづらいところが多いです。急に、「あれが○○だよ」と言われても詳しくない人は「そうなの？」という反応しかできないと思います。車の仕組みや走りは運転免許を持っていれば少しはわかると思うので、今回は形や種類、メーカーがわかるように書きます。

1. 最初はもちろん車の基本となる形。

形は**１ボックス**／**２ボックス**／**３ボックス**と呼ばれる箱の数によって分類されることが多いです。

**１ボックス**とはエンジンの上にドライバーが座るタイプで乗用車が**一つの箱**に見える車で、「トヨタ・ハイエース」がとても有名です。商業の車、に使われています。しかし最近は衝突などの安全性から短いボンネットがついたモデルが主流。

**２ボックス**とは乗員が乗るスペースの前にエンジンなどが入る**ボンネット**がついて**二つの箱**に見える車で、「マツダ・アクセラ」のハッチバックから「トヨタ・ヴォクシー」のミニバンまで幅広いモデルが当てはまる。

**３ボックス**とは乗員が乗るスペースの後ろに荷物を載せる**トランク**がついて**三つの箱**に見える車で、「トヨタ・クラウン」などが有名。また、タクシーや教習車はこのモデルが多いです。

ハイエース(１ボックス)　　　　　　　　　　　アクセラ(２ボックス/ハッチバック)

　　　ヴォクシー(２ボックス/ミニバン)　　　　　　　　　　　　クラウン(３ボックス)

1. 次は形からの目的別の種類。

種類としては、**軽自動車**、**コンパクトカー（ハッチバック）**、**背高ワゴン**、**ミニバン**、**セダン**、**ワゴン**、**四輪駆動（ＳＵＶ）**、**クーペ**、**オープンカー**、**エコカー**とさまざまな種類があり結構ややこしいです。

そこで下記では種類ごとに書きます。

**軽自動車**：最近は２ボックス。しかし、昔は小さいトランクがついて２．５ボックスの車もあった。規格からの制限があり660ccと小さいエンジンを積み、４人乗りと制限があります。燃費良く、維持費がとても安く、学生や街乗りに乗るにはとても良い。オープンカーからハッチバック、背高ワゴン、四輪駆動と幅広く存在している。

**コンパクトカー**：２ボックスの小型の車。1000cc～1800ccぐらいまでのエンジンを積み、５人乗りになる。そして街乗りから長距離ドライブなどに向いている。燃費も良いので軽自動車からの乗り換えや初心者などは良い。

**背高ワゴン**：２ボックス。上記のコンパクトカーと同じである。しかし、室内の高さがあり荷物などがたくさん積めるので、少人数旅行などでは最適。

**ミニバン**：２ボックスで室内は広い。３列シートにより６人から８人乗りとたくさんの人が乗る。これによって旅行、大人数のドライブなど最適。小型から高級感のある大型まで存在する。リヤはスライドドア(一部の車種)によって乗り降りがとても楽。

**セダン**：３ボックスの代表。４人が快適に乗れることや小型からスポーツタイプ、高級車はほとんど当てはまる。安全性、快適性は良い。

**ワゴン**：２ボックス。セダンの後席以降のルーフを延長させることで荷物がたくさん乗る。なので、セダンよりは使い勝手が良く、走りも良いモデルもある。

**四輪駆動**：２ボックス。一般道でも悪路でも快適に走れることができる。小型から高級感のある大型、３列シートを搭載しているモデルまで幅が広く存在している。

**クーペ**：３ボックス。セダンタイプの後ろの高さが低くなり２ドアになり、スポーツカーと呼ばれるモデルが多い。走りはとてもよくかっこいい。

**オープンカー**：３ボックス。クーペと同じであるが、屋根の部分が外せることによって解放感が抜群で、ドライブなどは最適。

**エコカー**：こちらは最近話題の環境対応車。

**ハイブリッドカー**：ガソリンエンジンとモータが組み合わさっている。モータがエンジンを助ける仕組みで必要に応じてモータのみで走ることができ、燃費が38km/リッターと世界最高水準となる。市販は多くのメーカーからされ、一番注目の車となっている。

**電気自動車**：ガソリンエンジンの代わりにモータが駆動力となりボンネットにはモータなどが収まる。また、タイヤに一つずつにもモータがついている。電気でバッテリーに充電するが走行距離がカタログ値で300kmと長い距離が走れないのが今後の課題。しかし市販がされ始めた。そして今年は一般の人も買える。

**プラグインハイブリッド**：これはハイブリッドのシステムに電気で充電できる仕組みを持つ車で、主役は電気自動車でバッテリーの充電がなくなれば、ガソリンエンジンが補助をする形。市販されているが一般の人はまだ買えない。

**燃料電池車**：これは電気自動車と同じ仕組みであるが、電気を水素と酸素の化学反応で作る車である。しかし機材が多く開発はされているが市販には相当かかると思われる。

1. 最後はもちろんメーカーはどんな車を売っているのか？

日本には**トヨタ**、**日産**、**ホンダ**、**マツダ**、**三菱**、**スバル**、**スズキ**、**ダイハツ**などとたくさんある。そんな中、このメーカーはどんな車が売れているか分かればいいと思います。しかし、あくまで独断。勝手ながら決めて書きます。

**トヨタ：**右のマーク。

**特徴**：愛知県豊田市に本社がある。

最近子供店長というCMで話題の日本を代表する車メーカー。もちろん生産台数も多い。

そしてダイハツ、スバルが傘下に入っているため今後は共同開発が多くなる。

　　　　そして世界初のハイブリッドカー「プリウス」もあり、環境への配慮も考えている。

　　　　そして、トヨタは右下の図の「レスサス」と呼ばれる高級車ブランドもあります。もともとアメリカで売られていたブランドで日本車では「セルシオ」が今では「レスサス・LS」として販売されている。

**有名車種**：「クラウン」「マークＸ」「カローラ」

「ヴィッツ」「ノア・ヴォクシー」

「ｂB」「パッソ」「プリウス」

「カルディナ」「オーリス」

　　　　　　　　「FT-86concept」が去年話題になった。

**日産：**右のマーク。

**特徴：**低燃費少女ハイジのCMやゴーン社長で有名ではないかと思う。

多彩な車や奇抜なデザインが多い。

**有名車種**：「マーチ」「キューブ」「ノート」「セレナ」

「エクストレイル」「スカイライン」「ＧＴ－Ｒ」

**ホンダ**：右のマーク。

**特徴：**バイクのカブで有名。

燃料電池車の開発やハイブリットカー「インサイト」の販売をする。

世界ではF１などのモータースポーツの参戦ではとても有名である。

箱根駅伝では車の提供などで有名。

**有名車種**：「フィット」「オデッセイ」「ステップワゴン」

「インサイト」「フリード」「ライフ」「ゼスト」

**マツダ**：

**特徴：**ZOOM－ZOOMなど、キャッチフレーズがあり、走りにこだわるメーカー。ロータリーエンジンの開発もしている。野球の広島の広告でもおなじみである。広島が本社。

**有名車種**：「デミオ」「アクセラ」「プレマシー」

「ロードスター」「ＲＸ－８」

**三菱**：

**特徴：**PC液晶などでも有名だと思うけど、車も作っています。

CMではdrive＠earthというキャッチフレーズがあり、走りと乗り心地にこだわるメーカー。世界ではラリー選手権「パリ・ダカ」で有名。

**有名車種**：「コルト」「ディンゴ」「パジェロ」

「ランサーエボリューション」「ｅｋワゴン」

**スバル**：

**特徴：**航空機で有名な富士重工業の車部門。

赤帽で使われている車では有名。悪路で走りをこだわる車づくりでほとんどが四輪駆動。エンジンも独自な車好きはファンが多い。

しかし、トヨタ傘下に入ったため、軽自動車部門は自社開発を終える。

今後はダイハツからの提供が入る。そしてトヨタと共同開発のスポーツカー「FT-86concept」のスバル版も今後販売予定。小型車も販売される。

**有名車種**：「インプレッサ」「レガシィ」「フォレスター」「ステラ」

軽トラの「サンバー」

**スズキ**：

**特徴：**静岡県浜松市が本社。バイクで有名。

軽自動車「ワゴンＲ」、軽トラ「キャリィ」も有名である。

**有名車種**：「ワゴンＲ」「アルト」「セルボ」「ジムニー」「スイフト」

軽トラの「キャリィ」

**ダイハツ**：

**特徴：**大阪府池田市が本社。

カクカク・シカジカのＣＭでおなじみ。そして多彩な軽自動車が多い。トヨタ傘下でトヨタと共同開発車の「ブーン」や「クー」がある。

また多彩なラインナップにより軽自動車の販売台数を伸ばしている。

**有名車種**：「ムーヴ」「ミラ」「タント」「コペン」

軽トラの「ハイゼット」

1. 余談

大体分かってもらえたと思います。しかし車はややこしいのは当然です。

あくまで自分解釈で書きました。

　※参考写真

ハイエース:「CORISM」

URL: <http://corism.221616.com/articles/0000065844/>

アクセラ:「livedoor ニュース」

URL: <http://news.livedoor.com/article/image_detail/4249838/?img_id=728901>

ヴォクシー:「Goo-net」

　　URL: <http://www.goo-net.com/catalog/TOYOTA/VOXY/10041826/index.html>

クラウン:「車っちょ」

　　URL: <http://www.kuruma-erabi.jp/toyota/crownsedan/>

トヨタマーク:「AUTOPARTSONLINE」

　　URL: <http://www.autopartsonline.jp/pg/maker/toyota.html>

レクサスマーク:「MeiwaSuisan」

　　URL: <http://bbs44.meiwasuisan.com/bbs/bin/read/car/1262680951/l50>

日産マーク:「楽天市場」

　　URL: <http://item.rakuten.co.jp/axis-no1/c/0000000254/?type=auction>

ホンダマーク:「Strange Blue」

　　URL: <http://strangeblue.cocolog-nifty.com/15is/2007/12/post-1e34.html>

リバースエンジニアリングの実践そのに　～日本語化～

**1** はじめに

\*ISM 第二回おめでとうございます。

というわけで連載との事で、第二回目、今回もまぁ適当にやってみましょうか。今回は前回と違い、やりながら書いていきます。分かりやすくはするつもりです。ちなみに、利用ソフトなどは前回と同じで

**OllyDbg**

OllyDbg

http://www.ollydbg.de/

日本語化パッチ

Digital Travesia ～ でじたる　とらべしあ～

http://hp.vector.co.jp/authors/VA028184/

です。

**2** 目標設定

前回、ゲームのバグ取りと正直そのゲームをやらない人にはどうでもいいネタでした。今回は、(少しは) 実用的です。

ターゲットは、Dependency Walker(ディペンデンシーウォーカーと呼ぶらしい) です。これ、Microsoft Visual C++ 6.0(VC6) とかについてくるツールで、一般公開されてます。Microsoft Visual Studio 2005 には付いて無いみたいですが…。さて、このソフトはEXE やDLL のインポート/エクスポート情報を解析するソフトです。Dependency=依存関係って事ですね。なので、自分で作ったプログラムが不要なDLLを取り込んでいないか、必要なDLL は何か、プログラム間のインターフェイスは正しく出力されているか、などを調べられます。他にも呼んでいるAPI で大雑把な目的を掴む、という用途も有ります。

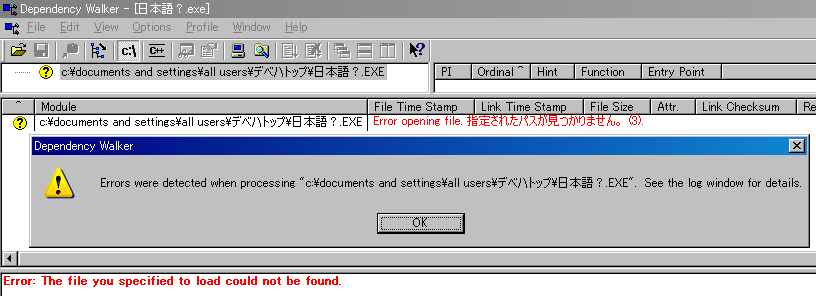
なかなかそういう作業には便利なのですが、いかんせん日本語対応が適当です。

図1: エラーパターン

というか、対応してません。日本語版のVC6 に付属してるくせに…ね。図1 を見れば分かりますが、「デスクトップ」をパスに含むと「デベハトップ」を参照して読めないと文句を言って終わりです。デスクトップで作業スンナといわれればそれまでですが、開発中のアプリなどはマイドキュメント以下に出来て、マイドキュメントはユーザフォルダ内部にあって、ユーザフォルダはユーザ名が日本語なら日本語で付いてしまうので、同じ現象がおきます。[[1]](#footnote-1)こうなるとに使えたもんじゃないですね。さて、長くなりましたが目標です。

Dependency Walker2.2 を日本語パスのファイルでも開けるように修正せよ！

**3** 原因特定

まずはデベハトップなどという面白単語に何故なったのか推測してみましょう。

デベハトップ以外の部分に注目すれば、小文字に変換している事が分かります。で、日本語文字(ShiftJisの場合) は、二バイト目に普通のアスキー文字と同じ値も入ります。二バイト目をアスキー文字として小文字に変えたり大文字に変えたりすると別の文字に変わってしまうと言うわけです。

良くある海外ソフトのお節介の弊害

1. 小文字に変えてしまう

デスクトップ→ デベハトップ

2. 大文字に変えてしまう

デスクトップ→ ェスクエィシ

3. Y(0x5C) 問題

ソ等を含むとそこにYが有るとしてパスが壊れる2

Shift-JIS の1 バイト目として有効な数値0x81 - 0x9F と0xE0 - 0xFC

Shift-JIS の2 バイト目として有効な数値0x40 - 0x7E と0x80 - 0xEC

この2 バイト目の0x40 - 0x7E が色々と他の文字と重なっちゃうわけです。

このY(0x5C) 問題は日本製でもたまに有ります。まぁ閑話休題

というわけで、日本語文字を無視して小文字に変換する部分をどうにかすれば、とりあえずこの問題は回避できそうです。

**4** 問題箇所特定

とはいえ、変換部分を直接探すのは難しいので、何かの指標から辿る事になります。ファイル名の変換を辿るので、ファイル名を取得する箇所からたどって見ます。ファイルの開き方も複数あるようなので、ここは簡単に、ドラッグアンドドロップから辿ってみます。これは、使われるAPI が(大抵は) 決まっているので、辿りやすそうだからです。OLE のドラッグドロップ関係を使われたらごちゃごちゃですが、幸い今回はそういうことはありません。

int DragQueryFile(Handle,index,buf,bufsize);

Handle で示されるドロップイベントの情報をbufsize 以下、buf に書き込む。index が-1 ならば、ドロッ

プファイル数を、index が-1 以外ならば、0 ベースで指定番号目のファイル情報を得る。

API の仕様(要約)

1. OllyDbg で読み込む。

2. 逆アセンブルペインで右クリック→検索→全ての外部関数呼び出し

3. DragQueryFile を探す。検索結果画面を関数名でソートして画面内に検索キーを入力すれば勝手にスクロールする。

さて、呼び出しを見つけたら、周辺を読みます。API の仕様からすると、ドロップファイル数をEBP-80に保存して、EBP-7C をカウンタとしたループで処理していることが分かります。で、ループごとにカウンタ番号目のファイル情報を0x104=260(MAX PATH) 文字有るらしいEBP-78 から始まるバッファに格納するようです。…なんだか、EBP-78 から0x104 進んだらえらい事になりそうですが、とりあえず謎の究明は置いておきます。で、読み込んだあとそれをパラメータにして…関数ポインタらしき構造が出てきました。(リスト1)

リスト1: 関数ポインタ？

1 00431C0A MOV EAX,DWORD PTR DS:[EDI]

2 00431C0C LEA ECX,DWORD PTR SS:[EBP-78]

3 00431C0F PUSH ECX

4 00431C10 MOV ECX,EDI

5 00431C12 CALL NEAR DWORD PTR DS:[EAX+88]

なんでワザワザ…と思うところですが、これはC++の仮想関数です。頭が痛いかたも、とりあえず深く考えないでください。

クラスの表現

どうしてこんな回りくどい表現なのか、順を追ってみて見ます。EDI がインスタンス(実体、クラス形変

数) で、EAX がvtbl とも言われる仮想関数の配列です。つまりリスト2 のようなコードが有った場合リ

スト4 のような形で処理されます。

リスト2: 仮想関数の例

1 class A{

2 public:

3 virtual void func();

4 };

5 A obj;

6 A.func();

リスト3: イメージ的にはこんな感じ

1 struct A{

2 void \\_func();

3 static const struct vtbl{

4 void (\*func)=\\_func;

5 } \\_\\_vtbl;

6 struct vtbl \*\\_vtbl;

7 };

8 struct A obj;obj.\\_vtbl=&A::\\_\\_vtbl;

9 A.\\_vtbl->func();

直接呼ぶ代わりにvtbl の関数ポインタを呼ぶわけです。

これを実際に仮想継承する場合はリスト4 がリスト5 のように解釈されるイメージです。

リスト4: 最終行ではA::func1 ではなくB::func1 が呼ばれる例

1 class A{

2 public:

3 virtual void func1();

4 };

5 class B:public A{

6 public:

7 virtual void func1();

8 virtual void func2();

9 };

10 B obj;

11 A \*p=&obj;

12 p->func1();

が

リスト5: A:: vtbl のように見えるが実際はB:: vtbl を呼んでいる例

1 struct A{

2 void \\_func1();

3 static const struct vtbl{

4 void (\*func1)=\\_func1;

5 } \\_\\_vtbl;

6 struct vtbl \*\\_vtbl;

7 };

8 struct B{

9 void \\_func1();

10 void \\_func2();

11 static const struct vtbl{

12 void (\*func1)=\\_func1;

13 void (\*func2)=\\_func2;

14 } \\_\\_vtbl;

15 struct vtbl \*\\_vtbl;

16 };

17 struct B obj;obj.\\_vtbl=&B::\\_\\_vtbl;

18 struct A \*p=(struct A\*)&obj;

19 p->\\_vtbl->func1();

これが仮想関数のオーバーライドを実現する方法(の一つ) です。

仮想関数なので、初期化するところを見れば呼ばれる関数も分かりますが、辿るのが面倒なのでブレークポイントを使って済ませます。仮想関数の使い方として、インスタンス(実体) だけ差し替えることで同じコードでも動作を変えられるというものがあります。それが適応されている場合には数回辿らないとダメですが…そういう小細工はされていないと仮定することにしまう。こういう手抜きにはそういう落とし穴も有ると覚えておきましょう。

ブレークポイントを仮想関数呼び出しに仕掛けて実行して、ファイルを読み込ませればOllyDbg に処理が移って停止します。

ここからはちまちま辿ります。ファイル名が何処へ移動・コピーされるか見失わないよう注意しましょう。総当りで解説しながら書くのもアレなので、勘に任せて探っていきます。使うのは「詳細ステップ」「ステップ」「リターンまで実行」です。それぞれ「関数などに潜る」「潜らないで1 行進む」「現在の関数の終了まで飛ぶ[[2]](#footnote-2)3」となります。

ですが本当に勘で探るのも面倒なので、メモリ検索を併用してさらに手抜きします。怪しげな関数に敢えて潜らず、ステップ実行で通過するごとに小文字に変換されたパスを検索することで、どのあたりで問題の処理が行われたか見つけられると言うわけです。検索するには、メモリウィンドウ開いて右クリックから検索します[[3]](#footnote-3)4。なお、検索キーはわりと短くても構いません。このソフトの場合、変換済みパスもウィンドウを閉じれば消えるようなので、変換される瞬間を見逃したらその前後にブレークを仕掛けてやり直します。関数を越えるごとに検索していけば「ある関数をステップで乗り越えると変換されたパスが出現する」というポイントがあるはずなので、次回はそこを詳細ステップしていく…と繰り返します。これを明確に変換している場所が分かるまで繰り返します。続けていると何度も呼ばれる関数の中で

0042994C CALL depends.004542BA

をまたぐ瞬間に変換されたテキストが出現し、この関数内部のリスト6 で示すループで変換されているところまで

たどり着けるかと思います。

1 004542EF /MOV CL,BYTE PTR DS:[EDX]

2 004542F1 |CMP CL,41

3 004542F4 |JL SHORT depends.00454300

4 004542F6 |CMP CL,5A

5 004542F9 |JG SHORT depends.00454300

6 004542FB |ADD CL,20

7 004542FE |MOV BYTE PTR DS:[EDX],CL

8 00454300 |INC EDX

9 00454301 |CMP BYTE PTR DS:[EDX],BL

10 00454303 \JNZ SHORT depends.004542EF リスト6: 小文字変換処理本体

見れば分かりますが、二バイト文字など考慮もしないで、ただただ大文字範囲の数値であれば小文字に変えるという処理を繰り返しています。

ちなみに、逆アセンブルペインの検索→コマンドのシーケンスに

CMP R8,41

ANY 8

ADD R8,20

と打てば、運が良ければ大文字→小文字の変換を一気に見つける事もできます。0x41 は大文字のA、0x20

は'a' と'A' の差です。なら最初からこれでやれ、というのは勘弁して下さい。

自分も正解を見つけてからその方法に気づくぐらいだったので…(汗

他にも「メモリ上で問題の小文字変換された文字列が現れる場所には規則性がある。よって、予めそ

こにメモリブレークポイントを仕掛けて書き込み処理を炙り出す。」といったアプローチもアリでしょう。

手段はたくさんあるので、このあたりは経験と勘と発想力次第です。

他のアプローチ

CMP R8,41

ANY 8

ADD R8,20

**5** 問題修正

さて、問題のコードは判りましたが、これをどうするかです。「同じDLL を二度検索しないように～」といった用途だったりした場合、そのまま潰してしまうのも多少不味いかもしれません。…まぁ深い事は気にしないで、とりあえず潰してみます。問題の加算命令をNOP に変えればいいですが、どうせなので変更量を減らしてみまましょう。

ADD の結果値が変わらなければ良いので、加算量をゼロにすれば無効変出来ます。

1. 問題の行をダブルクリック

2. 簡易アセンブラで20 を0 に変えて上書き

3. 変更した行を選択して、右クリックから「実行ファイルへコピー(選択行)」

4. 出てきた変更済みイメージを右クリックして、これを保存。

コピーの操作は実際にはコピーしないでパッチ済みイメージをメモリ上に作るだけなので、さらに保存する必要があります。ちょっと面倒ですね。

さてこれでテストしてうまく動いたのでおしまいです…ではあまりに無責任ですね。

というわけで、ちゃんとしたパッチも作ってみます。

今回は日本語対応が目的なので、日本語以外は視野に入れないパッチを作ります。

まず、どう考えても元の場所を上書きするだけでは足りないので、別の場所にコードを書くことにします。

インプラントとかパラサイトルーチンとか呼ばれます。このプログラムはコードがギッチギチに詰まっているようなので、終端の部分に埋めることにします。

リスト7: 解説不足のアセンブリ

1 st:

2 8A0A mov cl,byte[edx] ;Shift-JIS lead byte = 0x81...0x9F and 0xE0...0xFC

3 80E9 81 sub cl,81

4 80E9 1E sub cl,1e ; 0x9f - 0x81 = 0x1e

5 76 17 jbe sk ;if ck = lead byte : skip and next

6 80E9 41 sub cl,41 ; 0xe0 - 0x9f = 0x41

7 80E9 19 sub cl,1c ; 0xfc - 0xe0 = 0x1c

8 76 0F jbe sk ;if ck = lead byte : skip and next

9 80E9 45 sub cl,45 ; 0x41 - 0xfc = 0x45(overflow)

10 80E9 19 sub cl,19 ; 0x5a - 0x41 = 0x19

11 77 0C jnbe nx ;if ck != upper char : next

12 80E9 86 sub cl,86 ; 0x00 - 0x5a - 0x20 = 0x86(overflow)

13 880A mov byte[edx],cl

14 EB 05 jmp nx

15 sk: ;skip

16 42 inc edx

17 381A cmp byte[edx],bl

18 74 05 jz ed ;if next = NUL char : end

19 nx: ;next

20 42 inc edx

21 381A cmp byte[edx],bl

22 75 D5 jnz st ;if next != NUL char : loop

23 ed:

24 5B pop ebx

25 C3 retn

まずは埋めるべきコードを書いてみます(リスト7) sub の嵐ですね(笑)。…まぁジャンプの数を減らす小技です。

範囲の下限を予め引いておき、範囲の大きさを減算したときにキャリー(この場合正確にはボロー) が発生したら範囲内というわけです。最後、元の値に0x20 を加算する処理までsub でやってます。最後まで回ると確実にオーバーフローするのは…まぁアソビゴコロという奴です。ちなみに、加算でキャリーを作っても同じように作れる筈です。

このコードをEXE の開いているスペースに埋め込みます。まぁ適当に、0046C9A0 辺りが空いているので、そこから書いていきます。Jcc 系命令は相対ジャンプなので、範囲選択してバイナリコピー/貼り付けをすれば楽にコードを移動できます。この辺りが再配置性(リロケータブル) という奴です。たぶん。ついでに作業を楽にするため、0046C9A0 でラベルを貼ります。右クリックからでも「:」キーのショートカットでも構いません。

今回はaddcode とします。ラベルを貼って準備が終わったら、元のループの代わりにここへ移動するようにします。

つまり

004542EF MOV CL,BYTE PTR DS:[EDX]

004542F1 CMP CL,41

を

004542EF JMP addcode

と書き換えます。これで完成。先の簡易パッチと同じ手順でEXE ファイルに書き出します。保存するとき

は「全ての変更箇所」をコピーします。

jb とjl、ja とjg はどう違う？

簡単に言うと、符号つき整数用と、符号無し整数用の違いです。

jb、ja、jbe、jae は符号無し整数用で、jl、lg、jle、jge が符号つき整数用です。「IA-32 インテル(R) アー

キテクチャ・ソフトウェア・デベロッパーズ・マニュアル」の「命令セット・リファレンス」によると…

JA より上(CF=0 およびZF=0) の場合short ジャンプする。

JAE より上か等しい(CF=0) 場合short ジャンプする。

JBE より下か等しい(CF=1 またはZF=1) 場合short ジャンプする。

JB より下(CF=1) の場合short ジャンプする。

JG より大きい(ZF=0 およびSF=OF) 場合short ジャンプする。

JGE より大きいか等しい(SF=OF) 場合short ジャンプする。

JLE より小さいか等しい(ZF=1 またはSF<>OF) 場合short ジャンプする。

JL より小さい(SF<>OF) 場合short ジャンプする。

より上とより大きいの表現の違いがよく分かりませんが、CF やZF といった関連フラグは分かります。

CF はキャリー、OF がオーバ、ZF がゼロ、SF が符号(Sign) となります。ま、やってみた方が早いでしょ

う。

式ZF SF CF OF JB JL

0x01( 1)-0xFE(-2)=0x03( 3) 偽偽真偽真偽

0x01( 1)-0xFF(-1)=0x02( 2) 偽偽真偽真偽

0x01( 1)-0x00( 0)=0x01( 1) 偽偽偽偽偽偽

0x01( 1)-0x01( 1)=0x00( 0) 真偽偽偽偽偽

0x01( 1)-0x02( 2)=0xFF(-1) 偽真真偽真真

0xFF(-1)-0xFE(-2)=0x01( 1) 偽偽偽偽偽偽

0xFF(-1)-0xFF(-1)=0x00( 0) 真偽偽偽偽偽

0xFF(-1)-0x00( 0)=0xFF(-1) 偽真偽偽偽真

0xFF(-1)-0x01( 1)=0xFE(-2) 偽真偽偽偽真

0xFF(-1)-0x02( 2)=0xFD(-3) 偽真偽偽偽真

0x7F( \*)-0xFF(-1)=0x80(-\*) 偽真真真真偽

0x80(-\*)-0x01( 1)=0x7F( \*) 偽偽偽真偽真

JB はCF を見るので、減算の際に見えない上位桁から繰り下がると飛びます。

JL はOF が無い、普通に計算できる範囲の場合は結果が負になると飛びます。

そしてオーバフローを起こした場合は符号が反転してしまうのでOF との不一致…言い換えるとXOR を見て飛ぶようになっています。OF については加算で考えた方が分かりやすい概念で、引き算の場合は引く数の符号を逆にして考えたほうが理解しやすいでしょう。加算として考えて、「同じ符号の数値を足したのに符号が反転した」場合にオーバフローとなります。逆に符合が違う場合はどうやっても0 に近づく変化なのでオーバフローはしません。このオーバフローがおきた場合も正しく両者の関係を計るために、SF<>OF などといった訳の分からない条件が出てくるわけです。

**6** テスト

まずは簡易パッチの方から(図2)。

大文字がそのままで変換されていませんが、ちゃんと読み込めています。

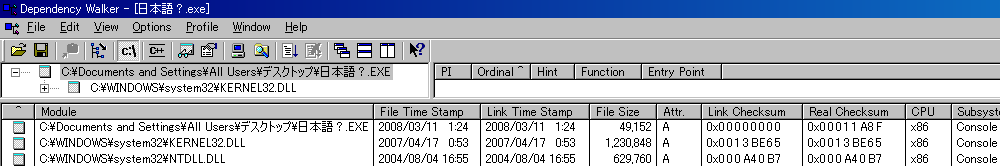


図2: 簡易パッチ

次、本命の日本語対応パッチの方(図3)。

ASCII 文字は小文字に変換され、日本語文字も壊れていません。

図3: 本命パッチ

大丈夫そうです。

というわけで、これにて終了！

**7** おわりに

今回はどうだったでしょうか。

なんだかコラム欄ばかり充実して内容が薄くなっている気がしないでも無いですが…理解できたでしょうか、楽しめたでしょうか。楽しんでもらえなかったならゴメンナサイ、楽しんでもらえたなら幸いです。

まぁ、リバースエンジニアリングやプログラミングの実践なんて、経験という名の「関連する雑多な知識からなる勘」が物をいう世界のような気もするので、コラムが多いというのもアリだと思います。…そういうことにして下さい。

最初に「やりながら書いていきます」と書きましたが、実は一つ前のバージョン2.1 でこのバグ修正はやっていたので、前提知識がなかったかと言うと微妙でした。とはいえ、なにをやったか内容を思い出せなかったので結局イチからやりましたし、そのぶん新鮮な気分で解析できました。終わったところで答えあわせを兼ねて古いファイルを調べてみたら…

……タイムスタンプが2006 年9 月1 日とかなってました。

…もう軽く二年も前だったのか…そりゃ忘れる……というか、ずっとこんな事ばっかやってたのか俺は。

9/10

リバースエンジニアリングの実践そのに　～日本語化～ 2008/03/12(2010/03/05 修正) 文月師走

今回は前回より実用的かつそれなりに応用の効きそうな内容に仕上がったと思うので個人的には満足し

ています。でわまた、機会があればリバースエンジニアリングネタをやるかもしれません。

以上ぅ！

PS. 今まで使っていた古いパッチは簡易版の実装でしたとさ。手抜きだなぁ…

PDAを用いたGPS信号の取得

昨今携帯電話やカーナビなどGPSを利用する機器がかなり普及し始めている。

そこで今回私が所有している東芝のPDA「GENIO　e550GX」を用いGPS信号を

受信しその情報から自分のいる場所を地図上に表示させてみることにした。

まず最初にGPSはどのようなものなのかの説明をする。

GPSとは「Global Positioning System」の略であり全地球測位システムと呼ばれるものである。

このGPSというものは元々米国が軍事目的に開発したもので、軍事用衛星から送られてくる

衛星の原子時計の時刻と衛星の軌道情報などを3個の衛星から受信し、受信機内の時計と照らし合わせ、情報が衛星から受信機までに届く時間に電波の伝播速度(約30万km/s)を掛けることにより衛星からの距離を割り出し、さらに3個の衛星との距離から一致する点が出てくる。

この一致した点が自分のいる場所になる。

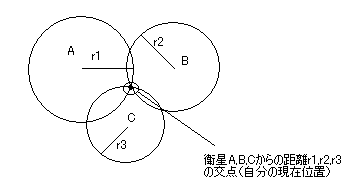


図１．受信点の求め方

衛星A,B,Cから受信点までの距離をr1,r2,r3とし、各距離を半径として

円を書くと各円の交点を求めることが出来、この交点の場所が受信点となる。

今回は2次元で求めたが実際は3次元で求めており、円ではなく球体により

位置を割り出している。

衛星から発信される信号には2種類あり、軍事用と民生用がある。

軍事用は非常に高い精度を

持っているが、民生用の場合は意図的に誤差が大きく出るようにしてある。

現在のところこの誤差は無効にしてあるが有事の際有効になる場合がある。

他に受信側の時計に誤差があった場合、受信点の場所がずれることがあるが、

最近のものでは4個目の衛星により正確な時刻を得て受信点を割り出すように

なっており、精度が向上している。

実際に動作を行うがここで使用するPDA、GPSのスペックを載せておく。

PDA東芝「GENIO e 550GX」

発売日　2002年11月15日

OS　 Pocket PC2002(Windows CE 3.0)

CPU Intel PXA250 (400MHz)

RAM 128MB

ROM 32MB

バッテリー　 リチウムイオン　1.100mAh

拡張スロット CFカード×１　SDカード×１

GPS 日本無線株式会社(JRC)

NNN-310TA

CF型

GPS信号より地図上に位置を表示するソフトはいくつかあり、

一部がフリーソフトで配布されている。

* アルプス社　モバイルアトラス　有料

PDAのバンドル品

地図が2002年のもので小域図が都市部のみしかなかったので

今回は使用せず。

GPS使用可

* 昭文社　Pocket Mapple Digital 有料

本屋にて3465円にて販売されていたため今回の会誌のネタおよびモバイルアラスからの地図の更新として購入。

GPS使用可

今回の会誌の記事のメインとして使用

* omani フリーソフト

フリーのソフトであり国土地理院の地図が使用可能で無料の地図ソフトになるが現在国土地理院で公開されていた数値地図(空間データ基盤)閲覧の試験公開が終了してしまい、無料で地図を

手に入れるのが難しくなってしまった。

Mappleの地図データを読み込ませることが可能。

GPS使用可

ほかにも数種販売もしくは公開されている。

PDAで使用する前に

最初にPDAの母艦となるPCにSuper Mapple Digital ver9をイントールする。

PCとPDAを接続(要ActiveSync)。

スタートメニューから「Pocket Mapple Digitalをインストール」選択。

インストール終了後Super Mapple Digital ver9を起動し、PDAで使用する地図データを

出力させてPDAへ転送する。

だいたいこの流れで使用することが出来るようになる。

地図の切り出し方などはSuper Mapple Digital ver9の起動時にチュートリアルが開くようになっているのでこれを見れば大体出来るようになる。

PDAにてPocket Mapple Digitalを起動し、出力した地図データを開くとしたのように

地図が表示される。(場所は切り出した場所によって異なる。)

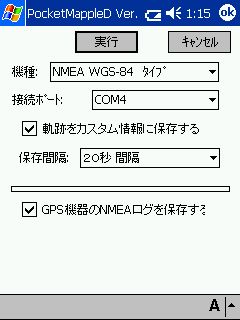


早速この地図に自分の位置を出力してみるがGPSを使うために設定をする必要がある。

設定方法

まず下に地図のタブがあるのでここを選択し、GPS位置取得を選択

すると下のような画面がでてくる。



ここの機種にあるWGS-84はGPS本体の測地系を表しており、

TokyoまたはWGS-84　のうちGPSの機種に合ったものを

選択しないとかなり大きな誤差が出てしまう。

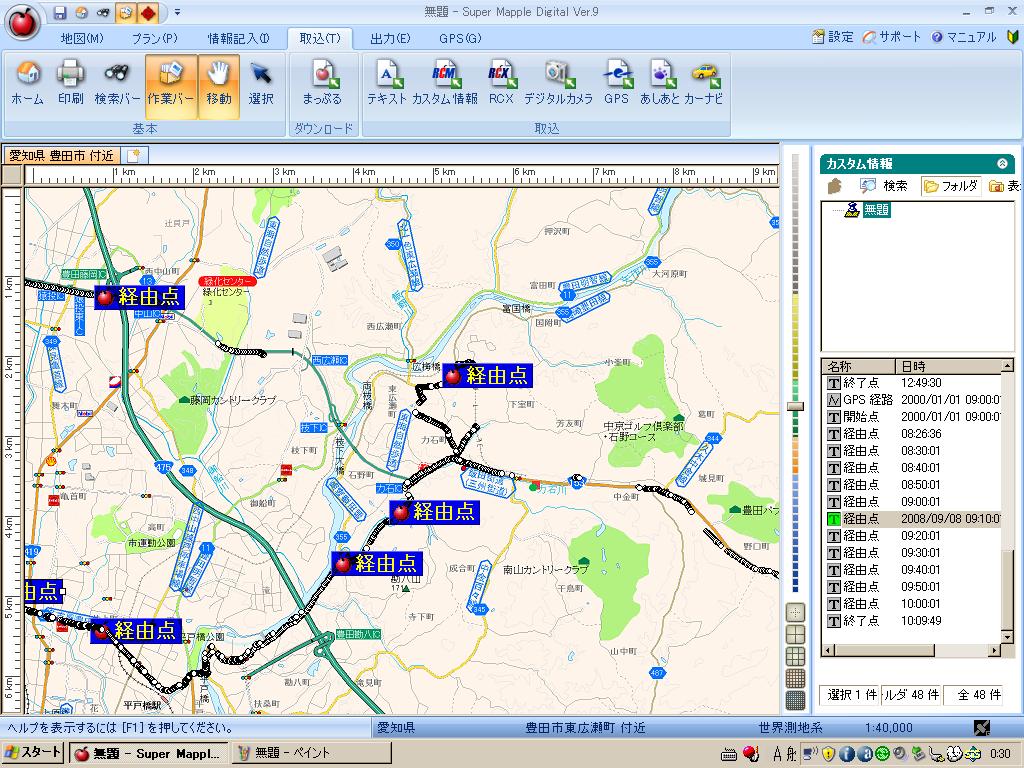
接続ポートはGPSユニットが接続されているポート選択するものである。

今回使用しているPDAではCOM4がGPSに接続されている。

これはPDAの機種ごとに違うので確認しなくてはならない。

GPS機器のNMEAログを保存しておけば自分が移動した軌跡を

Super Mapple Digitalに出力することが可能となる。



取り込みのタブにあるGPSでログを読み込ませれば地図上に自分の移動した軌跡が

表示される。（地図上にある白丸が奇跡である。）

ログをとる場合GPSの設定画面で実行ボタンを押すとログの保存先を聞かれる。



ここで保存場所を設定し、OKを押せば使用可能となる。

使用してみて思ったこと

　今回使用したPocket Mapple DigitalはPocketPC2002では動作保障対象外であり、

実際に使用するまでちゃんと動作するか不安だったがうちの2002ではほぼストレス

なく動作してくれたため非常に役に立ってくれた。

私自身バイクであっちらこっちら走り回るのでGPSのログをとり家に帰った後で軌跡を

たどってみると走った場所の近くに面白そうな場所を発見したりすることがあり、

次に行く場所の選定に役に立ってくれたり、軌跡を見るのも意外と面白い。

GPSを常時つけっぱなしになるためPDAの電池の持ちがかなり悪い

PDA自体が古いため電池が消耗している点もあるが大体1時間ぐらいで

30%以上食ってしまう2時間ちょいちょいでほとんどなくなってくる。

使用したGPSユニットが悪いのかGPS起動のたびに測位するまで福井県にある

日本無線の福井営業所を出力する仕様なっている模様。



そのためいったん電源を切るとこのような感じの軌跡ができてしまう。

アンテナの感度が悪いのか、測位するまで2~5分最悪10分以上

かかってしまう。

などがあげられる。

以上で説明を終了するがPDAによるGPSの利用はかなり使えることがわかった。

実際にとんでもない山道を走っている時に、この先ちゃんと道が繋がっているのか？

しっかりした道に出られるのか？と不安になることがあったがこのGPSにより自分の位置を正確に知ることができ非常に重宝した。

ここ最近携帯電話各社がスマートフォンを発売するようになったりPSPのGPSなど利用することの出来る機器が増えてきているので山へ行ったり方向音痴の方はぜひともこういったGPSを利用することをお勧めします。

以上で終わります。

4EE 川瀬

Open GL で図形を描こう

工学部電気学科情報通信専攻

2年 瀧本 将大

　パソコンで図形を描こう（OpenGLを使って）

1. 最初に

最近のPCゲームや家庭用ゲーム機（PS３とか）には、手に収まる携帯サイズのものも含めて本物と見紛わんばかりのCGが使われている。

　その中でも動かす為にPCに高い性能を要求するゲームには、ゲーム内の人や物の動きを実際にシミュレーションする物理演算エンジンを搭載したものもある。

ゲームではないがPhunという描画ソフトも有名で、中には自分の作った図形を組み合わせてロケットや丈夫な車（M1エイブラムス-イラク仕様-）を走らせる人もいる。

描いた絵や図形が実際に動いて、ソフトの中でシミュレートされるというのは凄い。

凄いが、ゲームの機能や物理演算エンジンが自分の描いた図形を動かす前は、

その図形はPCの中でどうやって扱われているだろう。(図形の情報をどうやって保持しているか)

　ここでは、OpenGLというAPI[[4]](#footnote-4)1(もしくはライブラリ)の支援のもとにC言語を用いてPCのウィンドウに図形を描いてみる。その際にOpenGLに偏ってしまうが[[5]](#footnote-5)2、C言語内でどうやって図形が覚えられるかを説明して、できた図形（の情報）へどんな操作をするとよりリアルになるのかを簡単に説明する。

　用語の説明やC言語が何をしているかを日本語で逐一書いていったりするのでC言語が分からなくても、あるいは開発環境がなくても雰囲気だけ掴むなら大丈夫・・・だと思う。簡単でわかりやすい説明とするあまり、実際の動きとは明らかに違うだろというのもあるかもしれない。それに冗長な為ある程度飛ばしてもいいかも。

今回の手順

　用語の説明　→　C言語の雰囲気というか掴み　→　C言語の上で動くOpenGL

→　OpenGLでの図形描写　→（図形を動かしたり画像貼ったり効果を付けたりする説明）

必要なもの（実際に図形を描くなら）

PC、開発環境(C言語のプログラミングができる環境)、GLUTのdll(ダイナミックリンクライブラリ)

・GLUTのdllを適切な場所に置く必要があるがそれにはglut.hやglut32.lib、glut32.dllという名前のファイルをネット上で探すのがいい。(適切な置き場所は開発環境の種類によって違うため説明しづらいが、最悪C言語で書いたソースコードの近くに置くか、ソースコードで参照する場所を指定すればいい)

1. 準備-用語の説明

CG・・・

コンピュータグラフィックス。PC（パソコン）の中で作られた画像。今回はこれを作る。

（マウスでいじると動くよ！）作り方は先程の通り、C言語上でOpenGLをガリガリと・・・

物理演算エンジン・・・

巨視的な系を古典力学的な法則でシミュレートするソフトウェア。

簡単に言うと、摩擦とか重さとか、衝突判定等をする。

Phun（ファン）・・・

マウスなどで描いた図形のオブジェクトを操作できる。

スウェーデン産の「物理演算お絵かきソフト」。

OpenGL（おーぷんじーえる）・・・

グラフィックインターフェース。描画の要。3DCGもできるが今回は2Dのみ。

図形の動作が命令一つで出るようにたくさん入ってる(?)

GLUT・・・

OpenGLにひっついて、OpenGLがやらないウィンドウの管理や

マウスの入力などをやってくれる。今回はこれを使っている。というかソフトが

OpenGLを使う際に餃子の皮のように包んでいるもの。

関数・・・プログラミング言語での処理の一区切り。

複雑な命令や内容をひとつの命令にまとめたもの。動き方は、

値を受け取り中身を実行して値を返す。中身を見なくても入出力さえ

理解できればブラックボックスとして使うこともできる。

ex)

「歩く」という名前の関数は「前の地面の状態」を受け取って

「前の地面に足を載せて…、重心をずらして…」という中身を実行し、

「歩けたか、歩けなかったか」を返したりする。

ライブラリ・・・

プログラムで良く使う命令をひとまとまりにしたもの。関数群。

上に例えると「歩く」や「泳ぐ」、「歌う」等が入っている。

C言語・・・今回の開発環境はVisual C2008

プログラミング言語。今年で38歳。PCは0と1しか分からないので、

その集合を人間が分かりやすい様にまとめた言語。だから命令の名前とかにも

英語がほとんど。書いた言語はスペルや構文のミスをチェックして

色々捏ねてから機械語(0と1による言語)に変換される（コンパイル）と完成。

今回はこれに描画支援プログラム(GLUT)を載せて動かす。

詳しくは次の方で。

ウィンドウ・・・

日本語だと「窓」だが、ディスプレイでアプリケーションに与えられる領域。

今回はCGが描画される範囲と同意。

オブジェクト・・・

いろんな意味があるが、今回は表示する図形の単体のこと。

1. C言語の雰囲気というか掴み

　先ほど説明したC言語は簡単な中身だと以下のような英語様な単語と記号の集まり[[6]](#footnote-6)3である。

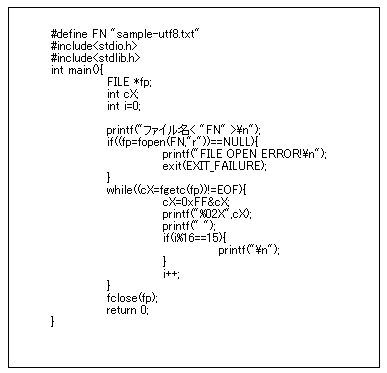


Fig.1

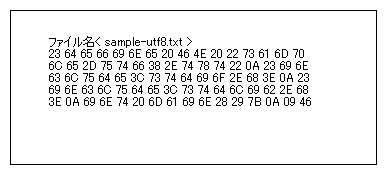
ちなみにこれは読み込んだファイルの中身を16進数で表示するプログラム。

Fig.2 　Fig.1をコンパイルして実行すると・・・

適当に指定したsample-utf8.txtというテキストファイルがこの様(Fig.2)に表示された。

ちなみにこの中身はバイナリエディタという16進数を表示できるソフトを使うと読める。

フリーで落ちているので読んでみると良いかもしれない。

1. C言語の上で動くOpenGL

Fig.1の様なC言語上にOpenGLで使われる関数をかいて実行する。この時、実際に書くのはFig.3でいうglClearColor()（背景の色を決める）の様に関数の名前や関数に引き渡す数値だけで、内部の処理は書かなくてもよい

（OpenGLで使えるとあらかじめ決まっている関数のみ。自作の関数は内部の処理も自分で書かないとダメ）。

1 #include<stdio.h> //ここでは、使用する関数が入ったstdio.h(標準出入力ライブラリ)などの

2 #include <GL/glut.h> //ファイルを参照するように命令している

3 #pragma comment(lib,"glut32.lib") //

4 void display(void) //ディスプレイに映す関数をこの中でまとめている。30行目

5 {

6 glClear(GL\_COLOR\_BUFFER\_BIT); //ウィンドウを塗りつぶす命令。25行目参照。

7 glBegin(GL\_POLYGON); //glBeginからglEndまでの間の描画点を

8 //[GL\_POLYGON]の指定する様に描く

9 //[GL\_POLYGON]：点同士を繋いで間の空間を塗りつぶす

10 glColor3d(1.0, 0.0, 0.0); /\*描かれる図形(頂点)の色をRGBで表している\*/

11 //この関数以降の描画点をR1.0G0.0B0.0 によりRedで描く

12 glVertex2d(-0.9, -0.9); //頂点の座標をx,y軸による座標で表している。

13 //(x,y)=(-0.9,0.9) 但し-1.0≦x≦1.0かつ-1.0≦y≦1.0

14 glColor3d(0.0, 1.0, 0.0); //この関数以降の描画点をR0.0G1.0B0.0 によりGreenで描く

15 glVertex2d(0.9, -0.9); //(x,y)=(0.9,-0.9)

16 glColor3d(0.0, 0.0, 1.0); //この関数以降の描画点をR0.0G0.0B1.0 によりBlueで描く

17 glVertex2d(0.0, 0.9); //(x,y)=(-0.0,0.9)

18 glEnd(); //図形描写の一区切り。8行参照

19 glFlush(); //まだ実行されていない命令(glBeginからglEnd)を、

20 //実際に画面に映すよう命令する

21 }

22 void init(void) //ウィンドウの初期化

23 {

24 glClearColor(1.0, 1.0, 1.0, 1.0); //ウィンドウを塗りつぶす色(背景)をRGBAで指定している

25 }

26 int main(int argc, char \*argv[]) //C言語はここから開始される

27 {

28 glutInit(&argc, argv); //GLUTの初期化処理

29 glutInitDisplayMode(GLUT\_RGBA); //ディスプレイの表示モード指定(GLUTによるRGBA)

30 glutCreateWindow(argv[0]); //ウィンドウを作成するように命令している

31 glutDisplayFunc(display); //displayを必要なとき実行する。図形はdisplayで描く。4行目

32 init(); //初期化処理のための関数をまとめるもの。22行目

33 glutMainLoop(); //display等をしながら終了を命令されるまで無限ループする。

34 return 0; //glutMainLoop()後にmain()が0を返すように命令している

35 }

Fig.3 OpenGLの関数が使われたC言語のソースコード

　なぜなら、よく使われる関数の中身はコンパイルするときに開発環境がOpenGLで使う関数を、関数名を参考に別の場所に参照しに行くからである（参照するライブラリ等を別の場所においておくのが面倒なときは実行するプロジェクトのソースファイル辺りに入れておけばいい）。

　Fig.3で言うと、ウィンドウを作成する前に使う関数を1,2,3行目のstdio.hやglut.hというファイル内から

取得している。その後、C言語の掟により26行目にあるmain()関数から順々に実行されてゆく。

内部の処理を追っていくと、まず28行目でGLUT/OpenGLの環境を初期化し、(絵具の準備)

glutInitDisplayMode()で描く表示モードを決定している(画用紙の選定)。

　そしてFig4の左上に見えるようなウィンドウを作成して、4から21行目の図形描画を31行目で命令している。init()ではFig.4のウィンドウの青い(白黒なのが残念)背景の様に初めに決めておけばいいことをまとめてあり、ここでの青い背景は32→22→24行目を辿って、glClearColor()が担当している。

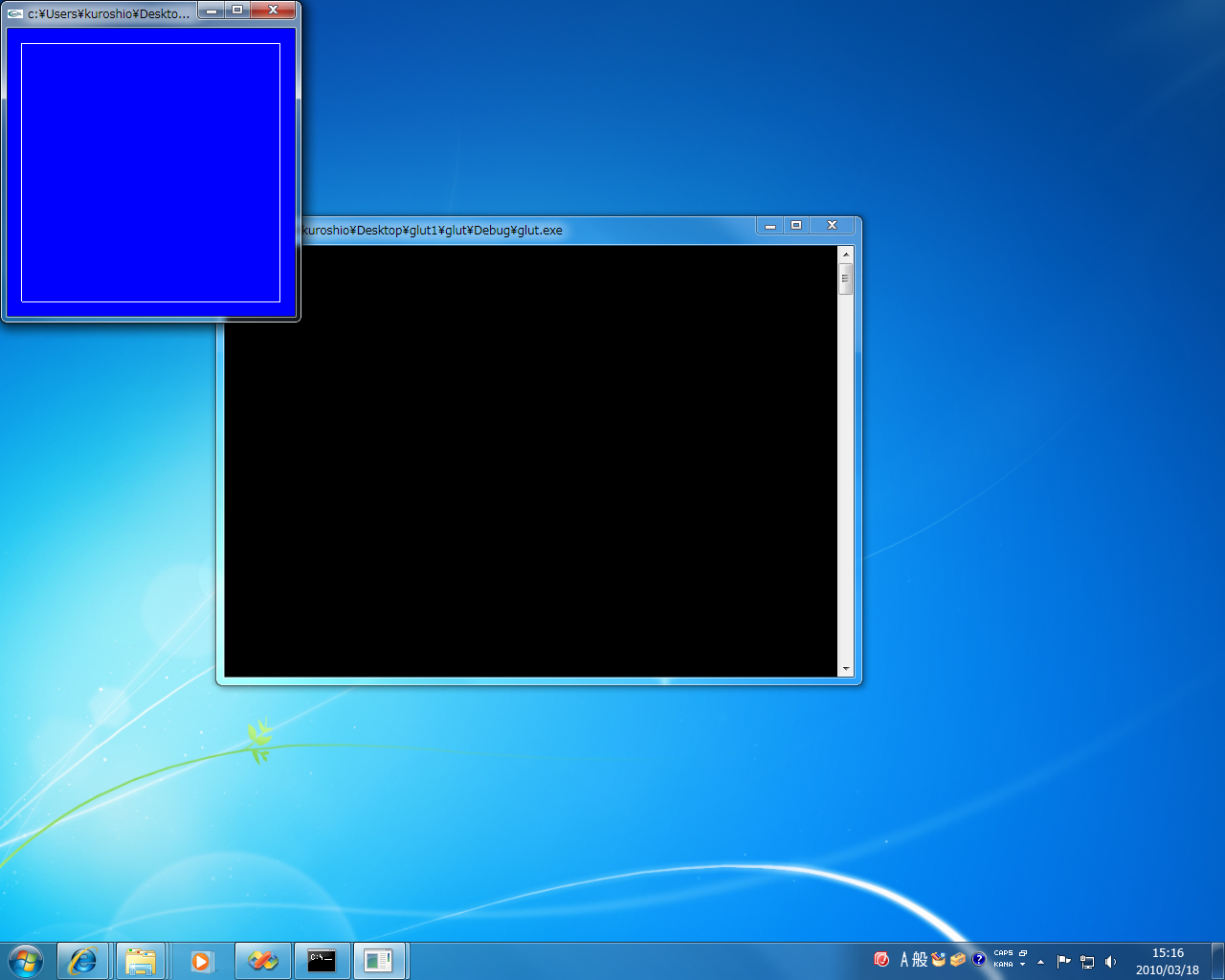
　そしてこの処理が終わると33行目のglutMainLoop()に処理が移る。この関数によってウィンドウが維持されておりこれが無いと一瞬で消えてしまう。本来この関数は、終了までの待機や、マウスやキーボードからの処理を待機するための関数である。

Fig.4　実行中の画面

1. OpenGLでの図形描写

実際の図形の形の話に移る。これは全部Fig.3の先ほど省略した31行目のglutDisplayFunc(display)

による4行目のdisplay()関数が担当している。まず6行目のglClear()がウィンドウを指定のバッファで塗りつぶし(決めた筆で背景を塗り)、7-18行目で書いた図形情報を19行目で書いたglFlushが一息にディスプレイに映している。

　7-19行目の処理は、glBeginで描き始め10行目のglColor3d(1.0,0.0,0.0)でこれより後の点を赤色で描くように指定している。次の行のglVertex2d(0.9,-0.9)でx=0.9,y=-0.9に一つの点を打っている。

　この点は7-19行目をみると3つありそれぞれが別の座標に打たれている。glEndまでの間に打たれたこれらの点が12,15,17行目の点の順に連結され、glBeginの引数(ひきすう)であるGL\_POLYGONによって

ポリゴンとして頂点と辺だけでなく面も描写される。線だけを描写したいならGL\_LINE\_LOOPを使えばいい。

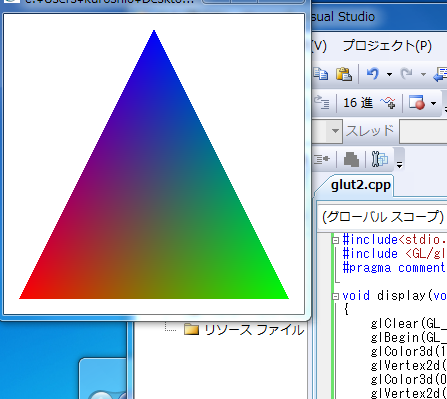
今回はそれぞれ頂点が違う色で指定されていたのでポリゴンとして表示すると、

Fig.5 カラーだとトッてもカラフル

このように表示される。

ちなみに、7行目や8行目のglで始まる関数がOpenGLの為の関数である。

今回のまとめ

　今回の記事では基礎知識的なことをどこに基準を置いて話すべきかがいまいち分からなかったので

また寄稿する機会があれば、今回を第一弾として次回以降にウインドウの制御や図形を動かす方法などを描いていきたい。

お世話になったサイト(環境を整える為に参考になりました)

Windows上でOpenGL

<http://www.02.246.ne.jp/~torutk/cxx/opengl/openglOnWindowsVC.html>

GLUTによる「手抜き」OpenGL入門　-床井浩平-

<http://www.wakayama-u.ac.jp/~tokoi/opengl/libglut.html#4.2>

OpenLDAP による Unix , Linux 総合認証システム

第１回

経営情報科学部 情報科学科 コンピューターシステム専攻

2TK 佐原 弘太郎

　\*ISM02発行おめでとうございます。当会のサーバ管理者として、今回は**非常に濃い**お話

ネットワーク・サービスの中の一つ、ディレクトリ・サービスの構築方法について記事を書かせていただきました。あり得ないほど長いですが、参考になれば幸いです。

0. 概要

　Unix / Linux 系では歴史的に複数台のアカウントを管理する際、NIS と呼ばれる認証システムが利用されていた。しかし、セキュリティ上の問題を抱えているなどの問題もあり、最近では後発の LDAP が使用されるようになっている。Windows 2000 以降のディレクトリサービスとして提供されている Active Directory においても、ユーザー情報の管理として LDAP を提供していたり\*1、Mac OS X 以降に搭載されている Open Directory は、ラッパーを経由して LDAP を使用している\*2。

　当文章では、LDAP によるユーザーデータの管理は二の次とし、POSIX アカウントの認証を行うことを主とする。

1. 作業環境及び仕様

1.1. 構築するOS と PC

　Windows Server 2003 Standard R2 上で実行されている　VMware Server 1.09 による仮想環境、以下の OS にて構築試験を行った。\*3

* VM Host Machine

PC : HP ML115 G5 ( Athlon X2 4450 2.4GHz , Memory : 4GB )

OS : Microsoft Windows Server 2003 Standard R2 SP2

* LDAP Server

OS : Debian GNU/Linux 5.0

Name : ldapsrv

1.2. 構築する LDAP 環境.

　当ドキュメントでは、LDAP にて連携可能である OS 全てを連携させるのが目的であり、特定のディストリビューションに固定されることのない構成を目指している。そのため、既存のローカルアカウントを移行することをせず、ユーザアカウントを独立して登録・管理する方式を取る。

* Domain : example.local
* Admin Passwd : hogehoge

1.3. 使用したパッケージ及びバージョン他

* OpenLDAP - slapd

2. LDAP の基礎知識

2.1 Lightweight Directory Access Protocol (LDAP) とは？

　ディレクトリ・サービスは、LANなどのコンピュータネットワーク上にあるユーザ情報、接続されているプリンターなどの資源を記憶し、検索しやすいようにまとめたものである。ネットワークを一元管理するための情報を保存し、利用するために、企業等の比較的規模の大きいコンピュータネットワークで利用されることが多い。

ウィキペディア（Wikipedia）- ディレクトリ・サービス より引用

　Directory とは、英和辞典によると「人名簿, 住所録」という意味となっている。元々、Directory Service というのは、人や建物を突き止めるためのサービス、ということになる。例えば、電話番号案内の 104 は、氏名・企業名・住所などから電話番号を引き当てる、立派な Directory Service である。

　しかし、コンピュータの世界に於いて「ディレクトリサービス」というのは、ネットワーク上に存在するコンピュータ・デバイス（プリンタetc）・ユーザーなどを管理するための機構のことを指す。DNS というのはドメイン名と IP アドレスを交互に結びつけるための\*4ディレクトリサービスであり、NIS は UNIX のアカウント情報を統括するためのディレクトリサービスである。

　このようにその用途のみに絞って設計・実装が行われているが、機器の種類も増え、リソースも爆発的に増大しているために、従来の「専用設計ディレクトリサービス」ではなく、ディレクトリサービスにアクセスするプロトコルも互換とし、登録出来る情報を拡張できるようにしたものが、 Lightweight Directory Access Protocol (LDAP) と呼ばれる設計である\*5。

　LDAP の設計概念上、数多くの情報が登録できるようになっている。強いて言えばネットワーク上に存在するリソース情報を一手に管理する、となるか。

* 電子的な電話帳 ( 名前、メールアドレス他 )
* アカウント / プロファイル管理.
* PKI 証明書の管理
* etc...

　現在の使用されている規格は、LDAPv3 と呼ばれるもので、SSL によるサーバ-クライアント間の暗号化、分散化の仕様など、大規模なネットワークでも十二分に効力を発揮するシステムとなっている。

2.2 LDAP のデーター構造

　LDAP のデーター構造は、大まかに物理的なものを Entry という仮想的なものに充て、論理的・組織的・物理的な階層ツリーに反映したものである（単純。

　※ 概念図

|  |
| --- |
| [root]  |  |- [Group1]  | |- (People1)  | |- (People2)  | |- (People3)  | |- (PC1)  | |- (PC2)  | |- (Printer1)  |  |- [Group2]  |- (People4)  | |- (PC3)  |- (People5)  | |- (PC4)  |- (PC3)  | |- (People6)  | |- (People7)  | |- (People8)  |- (PC4)  | |- (People9)  | |- (People10)  | |- (People11)  |- (Printer2)  [] が階層ツリーのグループ、() がエントリ。  実際、データー上はグループもエントリの一つとなる。 |

* **DN**

そのディレクトリサービス中において、エントリを識別する一意 (ユニーク) な名前のことを ''DN''((Distinguished Name - 識別名)) と呼ぶ。

* **RDN**

　そのエントリ自体の名前は **RDN**\*6 と呼ぶ。

2.4 エントリの属性

　各エントリは、以下の属性を必ず持つことになっている。

* objectClass

　このエントリが、どのような情報を持たなければならないかを定義する**オブジェクトクラス**。この識別子は**かならず設定されなければならない。**　このオブジェクトクラスの定義によって、設定することの出来る属性が変わってくる。主な ObjectClass を列挙する。

|  |  |
| --- | --- |
| Top | 基底クラス. このクラスが全てのオブジェクトクラスの基本となっている. |
| People | 姓名などを登録するクラス. |
| organizationalUnit | 組織を設定するクラス. |
| dcObject | ドメインを構築する |
| posixAccount | POSIX アカウントの属性クラス. ( /etc/passwd ) |
| shadowAccount | POSIX パスワードの属性クラス ( /etc/shadow ) |
| posixGroup | POSIX グループの属性クラス ( /etc/group ) |

この objectClass によって、エントリにどのような属性が付くかが異なる。例えば、posixAccount に属している場合、エントリの属性は以下のようになる。

|  |
| --- |
| dn: uid=people,cn=accounts,dc=example,dc=local  uid: people  sn: Marilyn Monroe  cn: Marilyn Monroe  objectClass: posixAccount  objectClass: shadowAccount  loginShell: /bin/sh  uidNumber: 1000  gidNumber: 1000  homeDirectory: /home/people  (表 2.4-1) |

　uidNumber , gidNumber , homeDirectroy , loginShell が、posixAccount のクラスに属する属性となる。

2.3 DN - Distinguished Name の形式

　LDAP相対識別名（RDN:Relative Distinguished Name）がカンマで区切られて並んでいるものが DN であり、そのデータベース上では唯一の名前となる。

　構造としては、(属性)=(値) が RDN であり、これらが連なったものが DN となる。

　属性は以下のようなものがある。

|  |  |  |
| --- | --- | --- |
| CN | commonName | 氏名 |
| L | localityName | ユーザの所在地住所(市/町) |
| O | organizationName | 組織名 |
| OU | organizationalUnitName | 所属部署名 |
| C | countryName | ユーザ所在地 (国名) |
| DC | domainComponent | ドメイン |
| UID | userid | ユーザID |

　実際はこの属性をすべて使うことは少なく、例のように

|  |
| --- |
| dc=example,dc=local |

や、

|  |
| --- |
| o=example,c=com |

　といった形式で、いくつかの属性のみしか使わないことが多い。

2.4 LDAP データ交換形式

　表2.4-1 で説明した際に説明した書式を**LDAP データ交換方式**と呼ぶ。LDAP のエントリをテキストで記述するための書式である。

|  |
| --- |
| dn: <識別名>  <属性記述子>: <属性値>  <属性記述子>: <属性値>  <属性記述子>:: <base64でエンコードした値>  <属性記述子>:< <URI>  (表 2.4-1) |

　複数行にわたり記述する際は、行頭がスペースまたはタブで開始すると前行に引き続き記述できる。

|  |
| --- |
| dn: uid=people,cn=accounts,dc=example,  dc=local  ↑スペース.  (表 2.4-2) |

　印字出来ない文字や、':' やスペースで始まる文字列の場合は、base64 にてエンコードして記述する。

また、属性値がファイルに格納されている場合は、以下のように URI で記述する。ftp , http にも同様に対応している。

|  |
| --- |
| cn:< file:///tmp/value  (表 2.4-3) |

エントリは一つのファイルに複数記述出来る。複数記述する際は、空行を挿入して他のエントリと区別する。

|  |
| --- |
| dn: cn=user1  <属性記述子>: <属性値>  <属性記述子>: <属性値>  dn: cn=user2  <属性記述子>: <属性値>  <属性記述子>: <属性値>  dn: cn=user3  :  : |

3. サーバ側の構築.

　今から行うことは、一通りパッケージをインストール・設定することにより LDAP の動作確認を行うものである。実際の運用では **LDAP サーバ上で LDAP と認証をリンクすることは非常に危険**であるため、最終的には外すと言うことにご留意戴きたい。

3.1. 必要なパッケージのインストール.

　まあ、さすがにこんなん構築しようと考えている人は分かると思うのでいろいろ省略。

ldapsrv:/root # aptitude\*7 install slapd

　　LDAP 管理用のパスワードの入力を求められるので入力しておいてね－。後で再設定するので適当でも良いと言えばいいｗ

　　LDAP サーバの再設定を行う.

ladpsrv:/root # dpkg-reconfigure slapd

|  |
| --- |
| Omit OpenLDAP server configuration?  => No  DNS Domain name: ( ドメイン名 )  => example.local  Organization name: ( ドメイン名 )  => example.local  Admin password ( 管理者パスワード )  => hogehoge  Database backend to use:  => HDB (Defalut)  Do you want your database to be removed when slapd is purged?  => No (Defalut)  Move old database?  => Yes (Defalut)  Allow LDAPv2 protocol  => No |

3.2. 設定書き換え

|  |
| --- |
| (追加)  include /etc/ldap/schema/misc.schema  include /etc/ldap/schema/openldap.schema  loglevel 0 → loglevel 256  access to \*  by dn="cn=admin,dc=example,dc=local" write  by \* read  ↓  access to \*  by dn="cn=admin,dc=example,dc=local" write  by self write  by users read  by \* read |

　また、ローカルに接続する権限を与えるため、/etc/ldap/ldap.conf ファイルに追記する.

|  |
| --- |
| HOST 127.0.0.1  BASE dc=example,dc=local |

　設定終了後、slapd を再起動する.

|  |
| --- |
| /etc/init.d/slapd restart |

3.3. slapd のシステムアカウントの製作.

　現パッケージでは自動的に openldap というアカウントを作成し、そのアカウント上で slapd が動作するが /etc/passwd を覗き見して openldap が存在しない場合は手動で作る必要がある。省略。

3.4. サーバ上に LDAP クライアントをインストール.

**ldapsrv:/root # aptitude install ldap-utils nscd**

　インストール完了後、以下のコマンドで LDAP Server ( slapd ) との接続試験を行う。

|  |
| --- |
| ldapsrv:/root # ldapsearch –x  # extended LDIF  #  # LDAPv3  # base <dc=example,dc=local> (default) with scope subtree  # filter: (objectclass=\*)  # requesting: ALL  #  # example.local  dn: dc=example,dc=local  objectClass: top  objectClass: dcObject  objectClass: organization  o: example.local  dc: example  # admin, example.local  dn: cn=admin,dc=example,dc=local  objectClass: simpleSecurityObject  objectClass: organizationalRole  cn: admin  description: LDAP administrator  # search result  search: 2  result: 0 Success  # numResponses: 3  # numEntries: 2 |

　と出力されていれば、初期設定完了.

3.5. 認証を LDAP に連携させる.

　取りあえずはサーバの認証を LDAP に結合してみる。

　nsswitch 用モジュールをインストールする。

**ldapsrv:/root # aptitude install libnss-ldap**

|  |
| --- |
| LDAP server Uniform Resource Identifier:  => ldapi://127.0.0.1  Distinguished name of the search base:  => dc=example,dc=local  LDAP version to use:  => 3 (Default ((slapd のセットアップで Ver 2 を禁止するとしたので 3 のみになる.)) )  LDAP database require login?  => No  make configuration readable/writeable by owner only  => Yes  Special LDAP privileges for root?  => Yes  Make the configuration file readable/writable by its owner only?  => Yes |

　後は、管理者を設定するのみである.

　続いて、nsswitch.conf を編集する.

|  |
| --- |
| passwd: compat ldap  group: compat ldap  shadow: compat ldap |

　passwd , group , shadow の項目に ldap と追加する\*8。

　次に、PAM 認証用モジュールをインストールする。

**ldapsrv:/root # aptitude install libpam-ldap**

　設定自体は libnss-ldap とそう大差ないので同じように設定。

|  |
| --- |
| LDAP server Uniform Resource Identifier:  => ldapi://127.0.0.1  Distinguished name of the search base:  => dc=example,dc=local  LDAP version to use:  => 3 (Default ((slapd のセットアップで Ver 2 を禁止するとしたので 3 のみになる.)) )  Make local root Database admin.  => Yes  LDAP database require login?  => No |

　後は LDAP の root を設定するのみである.

|  |
| --- |
| LDAP account for root:  => cn=admin,dc=example,dc=local  LDAP root account password :  => hogehoge |

　続いて、root パスワードの暗号化方式を指定する。 Unix / Linux 系のみの認証になるので md5 か crypt で問題無い。

　続いて、PAM の設定を行う。PAM とは Pluggable Authentication Modules と呼び、Unix / Linux の認証系の標準化？を目指して設計されたものである。認証はモジュール化されており、従来の /etc/passwd を担当する pam\_unix.so 等々、LDAP や ActiveDirectory , NIS と連携して認証が可能になっている。細かいことはググってください。

　Debian 系 ( Ubuntu とその派生を含む ) に置いては、/etc/pam.d/ に設定ファイルが保存されている。また、基本的な認証を行うモジュールに関しては、まとめて配置されている。

|  |
| --- |
| common-account  common-auth  common-password  common-session |

　以上のファイルにモジュールを追加すれば、その認証方法が有効になるという仕組みである。

* common-account

account [default=bad success=ok user\_unknown=ignore service\_err=ignore system\_err=ignore authinfo\_unavail=ignore] pam\_ldap.so

※ ↑全て一行で書いて。

* common-auth

|  |
| --- |
| auth sufficient pam\_ldap.so use\_first\_pass |

* common-password

|  |
| --- |
| password sufficient pam\_ldap.so use\_authok |

* common-session

|  |
| --- |
| session optional pam\_ldap.so  session required pam\_mkhomedir.so skel=/etc/skel/ umask=0022 |

　以上のファイルに指定された行を追加する。

3.6. 各コマンドと用語の説明.

　構築作業に入る前に、必要なコマンド群と用語を説明する。

3.6.1 コマンド群

|  |  |
| --- | --- |
| コマンド | 概要 |
| ldapadd | ネットワーク経由でエントリを追加する |
| ldapsearch | ネットワーク経由で エントリの検索を行う |
| ldapmodify | ネットワーク経由で エントリの更新を行う |
| ldapmodrdn | ネットワーク経由で RDN の変更を行う |
| ldapdelete | ネットワーク経由で エントリの削除を行う |
| slapadd | サーバ上のデーターベースより直接 LDIF 形式で登録を行う |
| slapcat | サーバ上のデーターベースより直接 LDIF 形式で出力する |
| slappasswd | パスワードの作成 |

　とどのつまり、接頭語として 'ldap' が付くのはネットワーク経由、'slap'と付くのはサーバ上のデーターベースを直接編集する = LDAP サーバ上でしか実行が出来ない！　ということになる。

　但し、slapadd に関しては **slapd を停止させなければならない**ことから、**通常時の使用は推奨されない**。エントリの追加には ldapadd を使用すること。

3.7 グループツリー及び、アカウントツリーの構築 ( ldapadd , ldapsearch )

　これからの作業は LDAP サーバ上 ( ldapsrv ) 上で実行する。

3.7.1 ツリーの追加 ( ldapadd )

**「ou=groups,dc=example,dc=local」**という DN の下に、POSIX でいう /etc/groups に相当する Group が登録されることになり、

**「ou=accounts,dc=example,dc=local」**という DN の下に、POSIX でいう /etc/passwd に相当する account が登録されることとなる。

　基本的な構造の構築を行う。以下のファイルを LDAP サーバ上に作成し、コマンドを実行する.

|  |
| --- |
| FileName : maketree.ldif  dn: ou=groups,dc=example,dc=local  ou: groups  objectclass: organizationalUnit  dn: ou=accounts,dc=example,dc=local  ou: accounts  objectclass: organizationalUnit |

**# slapadd -x -D "cn=admin,dc=example,dc=local" -w hogehoge -f maketree.ldif**

* -x : 簡易認証を行う ( SSL を使用しない )
* -D : 実行するユーザー ( DN ) を指定。
* -w : パスワード (平文)
  + hogehoge は admin のパスワード.

　各項目間は改行以外の文字を入れないこと、スペース他が入っていると正常に認識しない。

|  |
| --- |
| adding new entry "ou=groups,dc=example,dc=local"  adding new entry "ou=accounts,dc=example,dc=local" |

　と出力されれば成功した。ちなみに、もう一度同じコマンドを実行すると "ldap\_add: Already exists" となる。もう既に登録されているので、同じ名前では登録できないよ！　と言っているわけである。

3.7.2 エントリの検索 ( ldapsearch )

　正常に登録されたか、確認するために **ldapsearch** コマンドを使用する。

**# ldapsearch -x -D "cn=admin,dc=example,dc=local" -b "dc=example,dc=local" -w hogehoge**

* -x : 簡易認証を行う ( SSL を使用しない )
* -D : 実行するユーザー ( DN ) を指定。
* -w : パスワード (平文)
* -b : 検索を開始するツリーを指定する.
* ※hogehoge は admin のパスワード.

　と入力し、

|  |
| --- |
| (省略)  # groups, example.local  dn: ou=groups,dc=example,dc=local  ou: groups  objectClass: organizationalUnit  # accounts, example.local  dn: ou=accounts,dc=example,dc=local  ou: accounts  objectClass: organizationalUnit  (省略) |

　という結果が得られれば成功だ。その他、細かい書式に付いては省略させていただく。

3.9 各種コマンドの操作

　試しに一つエントリを登録し、それを蹂躙――もとい、いじくり回してみることにする。

3.9.1 エントリを１つ登録 ( ldapadd )

　LDIF 形式で追加するエントリを記述することにする。

* dn : ou=smith,ou=accounts,dc=example,dc=local

|  |
| --- |
| FileName : adduser.ldif  dn: cn=smith,ou=accounts,dc=example,dc=local  objectClass: top  objectClass: person  cn: smith  cn: John  sn: Smith  (表 3.9.1-1) |

**# ldapadd -x -D "cn=admin,dc=example,dc=local" -w hogehoge -f adduser.ldif**

　無事登録されれば、adding new entry ほげほげ となるはずである。

3.9.2 エントリを編集してみる ( ldapmodify )

　LDIF 形式で編集するエントリを記述する。LDIF 書式で記述するのがめんどくさいのだが、慣れればかなり高度なことが出来るようになる。

|  |
| --- |
| FlieName : modifyuser1.ldif  dn: cn=smith,ou=accounts,dc=example,dc=local  changetype: modify  add: cn  cn: john |

**# ldapmodify -x -D "cn=admin,dc=example,dc=local" -w hogehoge -f modifyuser1.ldif**

3.9.4 エントリを検索する ( ldapsearch )

　エントリを検索する場合は、ldapsearch で検索する。

**# ldapsearch -x -b 'dc=example,dc=local' -s sub '(cn=\*smith\*)'**

　正確な書式は省略するが、**-b** オプションで検索する階層を指定、**-s sub** オプションで下層のエントリを検索、最後の引数が検索式となる。

　細かいことは、**man ldapsearch** 。

3.9.5 エントリを削除してみる ( ldapdelete )

　ldapdelete のコマンドの引数に、削除するエントリの DN を指定する。

**# ldapmodify -x -D "cn=admin,dc=example,dc=local" -w hogehoge -w 'cn=smith,ou=accounts,dc=example,dc=local'**

　確認のために、前項の ldapsearch で cn=smith,(略) を検索すると、存在しないことが確認できる。

4. POSIX アカウント

4.1 POSIX アカウントのクラス

　POSIX のアカウントを登録するには、いかにあげるいくつかのオブジェクトクラスを必要とする。

* posixAccount
* shadowAccount

　objectClass が posixAccount である場合、以下の属性識別子が存在する。

|  |  |
| --- | --- |
| uid | ユーザーの表示名 |
| uidNumber | このアカウントのユーザーID |
| gidNumber | このアカウントが属するグループID |
| homeDirectory | ホームディレクトリの指定 |
| loginShell | 使用するシェル |

　objectClass が shadowAccount である場合、以下の属性識別子が存在する。

|  |  |
| --- | --- |
| userPassword | 暗号化されたパスワード |
| shadowLastChange | 変更日時 |
| shadowMin | 変更不能日数 |
| shadowMax | 変更要求日数 |
| shadowWarning | 期限満了警告日数 |

　これらを二つを組み合わせてユーザー情報を作成するために、サーバ上に以下のファイルを作成しコマンドを実行する。

* 新しく作成するユーザ名 = uid は hoge
* uidNumber は 10001 , 所属する gidNumber は 10001.
* ホームディレクトリは /home/hoge
* 使用するシェルは /bin/bash
* パスワードを hogehoge とする

|  |
| --- |
| FileName : adduser\_hoge.ldif  dn=hoge,ou=people,dc=example,dc=local  cn: hoge  uid: hoge  uidNumber: 10001  gidNumber: 10001  homeDirectory: /home/hoge  loginShell: /bin/bash  userPassword: {MD5}MpQ15eZr6AmmVq8QX0JAHg== |

**# slapadd -x -D "cn=admin,dc=example,dc=local" -w hogehoge -f adduser\_hoge.ldif**

　userPassword を見ると {MD5} となっている。これは、暗号化の方式を表すものであり、暗号化されていることを示す。もちろん、平文で **userPassword: hogehoge** と書くことも可能だが、セキュリティ上よろしくないので暗号化することをおすすめする。

　暗号化されたパスワードを作成するためには

|  |
| --- |
| # slappasswd -h '{(暗号化方式)} |

　と実行すると、パスワードの入力を求められるので、暗号化するパスワードを入力する。

　暗号化の方式はいくつかあり、

* CRYPT
* MD5
* SSHA

　などがあるが、暗号の強度的に MD5 か SSHA をおすすめする。

4.2 POSIX グループのクラス

　　POSIX でグループを作成する場合、**posixGroup** と呼ばれるオブジェクトクラスが必要となる。

　属性識別子は単純なので、直接 LDIF で書いたものを見てみることにする。

|  |
| --- |
| FileName : addgroup\_human.ldif  dn: cn=human,ou=groups,dc=example,dc=local  objectclass: posixGroup  cn: human  gidNumber: 10001 |

　cn が Group 名( **human** )、gidNumber がグループ番号( 10001 )となる。

　上の内容でサーバ上にファイルを作成し、以下のコマンドを実行する。

**# slapadd -x -D "cn=admin,dc=example,dc=local" -w hogehoge -f addgroup\_human.ldif**

　以上、これでアカウントの登録が完了した。

4.3 アカウントの試験

　サーバ上で、以下のコマンドを実行する。必ず、一般権限ユーザで実行すること。root 権限だと、su はアカウントが存在する場合、無条件でそのアカウントに移行するため。

$ **su hoge**\*9

Password: **hogehoge** \*10

　これでログイン出来たならば、PAM の設定・LDAP での認証が成功しているとなる。

　次回は、実際に運用するための Tips となる予定である。次回をお楽しみに！（ぉ

\*1 Active Directory では認証自体は Kerberos 認証が受け持っている

\*2 Open Directory に置いても認証は何故か Kerberos 認証…… AD との連携を重視したのだろうか

\*3 Debian 5.0 上で試験しようかと思ったが VMware Server のコンパイルが通らなくてイラッとしたので取りあえず間違いなく動く Windows Server 上での試験となった

\*4 一方方向になっていることも多々あるが……

\*5 本当は LDAP が出来る前に、OSI の X.500 という設計があったらしいが、筆者はよく知らない

\*6 「Relative Distinguished」の略。

\*7 apt-get でも可. 常日頃使用している方を利用して下さい.

\*8 compat が files である可能性はあるがどちらでもかまわない。

\*9 試験で作成したアカウント

\*10 hoge アカウントのパスワード

P.S.

いろいろ面倒な記事で編集さんごめん！

工科展作業班の報告。

今回も眩しいくらいにプロジェクトが燃え盛りました。

Make04になんとか間に合ったのが奇跡としか言いようがないというのも開発開始の遅れと人的資源の割り当てのミスが行程の遅延を発生させ致命的な結果を招いたためです。

何かやるには技術も必要ですが技術とは必要にならないと身につきにくいものだったりします。お恥ずかしながらプロジェクトの管理などの技術は身に付きませんでしたがFPGA回りやアナログ回路周りの一部の技術を身につけることが出来ました。

課題も多かったですが、複数人でチームを組んでシステム開発を行うことは良い経験となりました。

**工科展主任 - 2TK 畔柳隆敏**

０９年度工科展主任をしました畔柳です。

今回は、あの楽器ってことでsm5480792←これを作ってみるってことで

作ってみた結果がMake04で展示したあれです。

まあ、相変わらずプロジェクトは燃え上がるものですね。

工科展とか余裕で間に合いませんでしたからね。

区分切ったのはいいけど人の配分に問題が合ってハードウェア周りから燃え上がりました。

今回の肝である検出機構周りは私がずっと考えていて（後から聞いた話ですがこの機構の特許とってるとこあるらしい）素子間の遮蔽物の有無の情報から物体の位置を検出する機構を考案し、設計しました。

ここで高速な計算を必要とするため固定論理で処理を行ったりするためにFPGAが、素子の受光素子側で増幅が必要なためにオペアンプが必要となりアナログからディジタルまで幅広い知識とFPGAで使用する開発ツールが必要となりました。

また、発光素子の制御に汎用ロジックICを使う予定をしておりそのように設計を行ったところ非常に多くの石を使い消費電力も相当なものとなることが判明し、部品選定からやり直す事態が発生したのもプロジェクトに大きな影響を与えました。

最終的に発光素子はCPLDとトランジスタアレイによる制御となり消費電力の問題などは低減されることとなりました。

ハードウェア側の遅延は致命的なレベルで、１０月の工科展には間に合うことが出来ませんでした。Make04に間に合わせることが出来たのは幸運としか言いようがないくらいに奇跡的なことでした。

ソフトウェア側は、midiサーバとネットワーク周り以外問題なく行程をこなしてそれなりのものができました。しかし、UIのチューニングなどはハードウェアが存在しないことで出来ずその辺りで不満が残るものとなりました。

このような組み込み系のシステムの開発の場合にはソフトウェアはハードウェアの仕様や癖により大きく左右されることも

あるのでハードウェアの先行設計を行わなければ不十分だったのだと考えています。

特に回路設計やMPU,プログラム可能LSIなどを利用する場合には幅広い知識が必要となるため場合によってはリファレンスを読みつつ開発を行うということを行わなければならなくなります。今回の受光回路の変換増幅回路やFPGAでの固定論理演算機などはそのような開発スタイルをとりました。

私が不甲斐ないばかりに他メンバには大きな負担をおかけし大変申し訳なく思っていますが、複数人でチームを組んでのシステム開発で特に主任という人をまとめる立場に立ったことは非常に良い経験となりました。

ここで多くの方々に感謝と謝辞を申し上げます。

どうも1年間ありがとうございました。

**技術主任 - 2TK佐原**

“あの楽器”のインターフェイス設計・MIDI機器の制御プログラム・ネットワークの通信プログラムから、物理配線まで担当した貧乏器用な佐原です。なんでもできるっていいことだよ。やることが詰んできて大変だけどな！

みなさん、若いうちにデスマーチをしておきましょう。きっと、もしかしたら、たぶん…………強くなれます。

Let’s デス・マーチ。

**インターフェイス主事 – 2EJ瀧本**

教訓。使えないやつは早く切り捨てましょう。

**インターフェイス主事 – 2EJ石川**

**インターフェイス主事 – 2EJ倉地**

今回、インターフェイス主事担当と広報担当となり、インターフェイスではOpenGLをほとんど知識がない状態からの作業で戸惑いもあり迷惑をかけたと思います。広報では工科展ブログ（途中で…）やポスター作りをしました。

　　こつこつ作業をやっていくことは大切です。

　　　MTM04出展時には**ムーヴ**で**東京**へ行ったことが最大の行動！

**筺体主任 – 2MM戸澤**

役員からの一言

会長の瀧本です。

総務

書記

会計の畔柳です。

最近会計仕事しろと言われそうでビクビクしております。

今年は部室の移転などがあるようでサークルの一つの節目の時期となりそうです。

備品の更新や新規購入などで今年は出費が多くなりそうですので財布の紐はしっかりしめていきますが、必要なものやあったらいいと思うものは積極的に総会で提案してみてください。良さそうならなるべく要望が通るように努力しますから。

初めに会計に話を通しておくと通るとかはないですが、

年間での収支は保ちたいので調整が可能かもしれません。

みなさん。では

くろやなぎ でした。

編集後記「会誌としては私の入学して以来二度目となる本誌ですが、昨年の合宿や工科展などの行事を経験してシス研の空気を肌で感じた会員から寄せられた記事を見るとこの会誌も一つづつの行事を継いだひとつの行事なのかなと感じました。」

シス研への連絡方法

[contact@sysken.net](mailto:contact@sysken.net)

<http://aitech.ac.jp/set1/>

1. 実際の名前は英数字だけどローカライズされたためエクスプローラでの表示は日本語ってケースも有ります [↑](#footnote-ref-1)
2. 3 若干不安定な機能ですので、上手くいかないときは素直にブレークポイントとステップ実行で処理してください [↑](#footnote-ref-2)
3. 4 Ctrl+B でも検索ダイアログを出せます [↑](#footnote-ref-3)
4. 1 アピ。OSと表面で動くソフトの仲介屋。そのOSでソフトがよく使う機能を纏めたもの。 [↑](#footnote-ref-4)
5. 2 DirectXを用いる事もあるが、筆者の知識の偏りとハードに左右されにくい汎用性、文献の多さによりOpenGLを採用した。 [↑](#footnote-ref-5)
6. 3 これだけでも意味は伝わるが、普通は読む人のためにコメント文という注釈を載せる。 [↑](#footnote-ref-6)