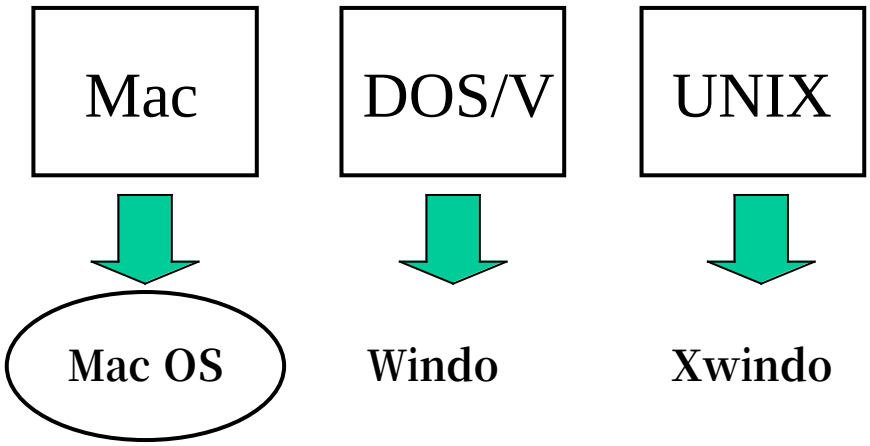
1　実験の概要

1.1　　Xwindow について

Xwindow は、UNIX のワークステーション上のウィンドウシステムでもっとも一般的なものである。ウィンドウシステムとは、スクリーンをウィンドウという領域にモジュール化し、文字や図形・画像をふんだんに使った、人間にとってわかりやすく、使いやすく、習得しやすいインタフェースである。その本来の機能は、グラフィクスを使った人間とコンピュータの間の直感的なインタフェース、すなわち”GUI(graphical user interface)”という点にある。わかりやすい例で、Apple Macintosh の Mac OS や DOS/V マシンの Microsoft Windows などが挙げられる。



この、ウィンドウシステムには、様々な機能があって、目的に応じて多種多様の用途がある。今回、本実験の主旨は Xwindow で、”アプリケーション・プログラミング”をすることである。アプリケーションとはマシンや OS、ライブラリの性能を試すため、あるいは実際の目的に”応用”するために作られたプログラムのことを言う。実際のところプログラミングをする際に、自分のプログラムがアプリケーションであるのかないのかなどという難しいことは考えることは必要なく、Xwindow でプログラムを作ると、それは必然的にアプリケーションと呼ばれるだけの性質を持つことになる。つまり”Xwindow でプログラムを作ること”こそが、本実験の目的といえる。

また、本実験で Xwindow System を使用し、UNIX のコマンドやプログラム等々を行なうことによって、UNIX に少しでも慣れ親しんでもらおうという趣旨も、たぶんにふくまれている。

1.2　演習問題「時計の作成」

1.2.1　処理の流れ

Xwindow のプログラミングを用いて、新しくウィンドウを生成し、その中にアナログ時計を表示させるプログラムを作ります。
処理の流れは、次のようになる。

① 新しく、ウィンドウを生成する。
② システム中から、現在時刻の情報を取り出す。
③ 取り出した時刻をもとに、秒針・分針・時針それぞれの角度を計算する。
④ 生成されたウィンドウの中計算された角度をもとに、円や直線などを描くことによりアナログ時計として表示させる。

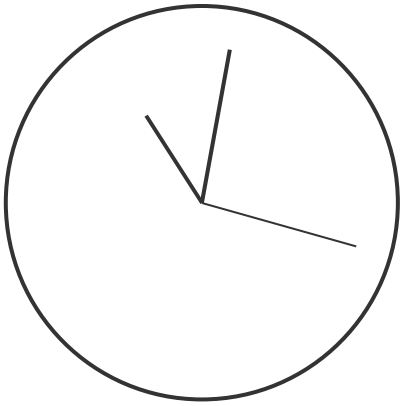


図.時計のイメージ

1.2.2　　サンプルプログラム

今回用意したサンプルプログラム「 sample.c」は、現在時刻の時間・分・秒を表示させるのと、ウィンドウの中に円と線を描いたプログラムである。このプログラムを自分のディレクトリの中に別の名前でコピーし、そのコピーされたファイルを実験用とし書き換える。（コピーの仕方は、後に参照）

ここで、サンプルプログラム中の関数を簡単に説明する。

- win__set(x,y)

main プログラムの中で、この関数を呼んでいる部分があるが、今回の実験ではこの関数の中身を書き換える必要は特にない。新しくウィンドウを生成する関数で、出力されるウィンドウのサイズなどを定義している。時間に余裕があるものはこの関数を書き換えて考察の 1 部に加えてみるのもよい。本実験では、この winset によって生成されたウィンドウに、時計を表示させるという仕組みである。

- timer(&hh, &mm, &ss, &ssb)

この関数は、システムから時間の情報を取り出し、時間の値を hh、分の値を mm、秒の値を ss にそれぞれ格納する。そこで、main プログラムの中で、秒針・分針・時針を描く場合現在時刻のそれぞれの情報は、ss, mm, hh を使えばよい。

- Clock_display(hh, mm, ss)

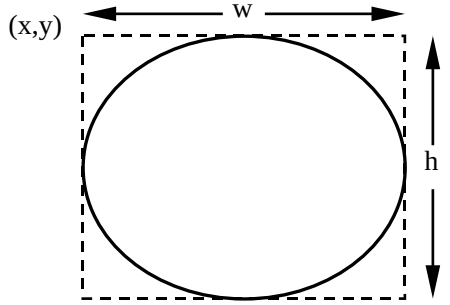
アナログ時計を表示させるには、ウィンドウ上に円や線を描かなければならない。その、円や線を描くのがこの関数である。この中で線や円を描くための関数を呼んでいるので、それらについて説明する。

  - XSetForeground(disp, gc, 0)
                                a

    色を指定する関数です。下線 a の数値を変えることにより、色を指定することができる。

  - XDrawArc(disp,win0,gc, 25, 25, 150, 150, 0, 360*64)
                                x y w h

    楕円を描きます。x, y w, h はそれぞれ下の図のような寸法を示している。真円を描く場合は、w, h の値を同じにしなければならない。



■　XDrawLine(disp,win0,gc,center,center,center,center+100)
x_1 y_1 x_2 y_2

線を描きます。x_1, y_1 は始点座標、x_2, y_2 は終点座標をそれぞれ決めている。

サンプルプログラム中では、ウィンドウ上にただ円と線を描いているだけである。この Clock_display 関数には、timer 関数によって取得した時間・分・秒の情報(hh, mm, ss)が受け渡しされているので、その時・分・秒の情報を元に時針・分針・秒針の角度を計算して、時刻の経過とともにそれぞれを円の中に描くことによって、時計が完成する。ここで注意する点は、プログラム中での角度の単位は、ラジアンであること。

次に、プログラム中で指定している変数について、説明する。

center	ウィンドウの中心
hh_needle	時針の長さ
mm_needle	分針の長さ
ss_needle	秒針の長さ
pai	πの値

もしほかに変数が必要ならば、初期設定の所に必要な型で宣言すること。

サンプルプログラムは
make
でコンパイルを行うと「sample」という実行可能なファイルができるので
sample
(デバッガで実行するときは run)
で実行する。

サンプルプログラムをコピーし、実験用のファイルを作る。
cp sample.c ファイル名.c
エディタを立ち上げ（emacs 等）実験ファイル用に Makefile を書き換える。
file = sample ⟶ file = ファイル名

以下の順序をたどって、プログラムを作成する。

１．　ファイル名.c をエディタで立ち上げる。
２．　プログラムを書き換える。
３．　make でプログラムをコンパイル。
４．　コンパイルが成功すると実行可能形式のファイルできるのでファイル名で実行

＊コンパイルでエラーが出たり、実行がうまくいかないことがあれば、
１．にもどる。

表示された画面は xwd（X Window Dump）**というコマンドでファイルに取り込むことができる。コマンドの使い方は、**
xwd > **ファイル名**.xwd
とすると「ピッ」と鳴るので、その後取り込みたいウインドウをクリックする。

さらにこれを convert **で** PS(PostScript)**ファイルに変換することで印刷可能となる。使い方は**
bin/convert **ファイル名**.xwd **ファイル名**.ps
さらに印刷は、
cat **ファイル名**.ps | rsh s02 lpr
この方法を用いてレポートに添付する実験結果を作成すること。
＜注意＞*.xwd **を印刷**（lpr）**しないこと！**

2　レポートについて

2.1　　作成方法

レポートは以下のような構成にする

1.表紙

別紙に添付した表紙を利用すること。

2.実験の背景

3.　　アナログ時計のインプリメント（プログラム化）

実際にどのような仕組みでプログラムが動いているのかを説明する。実際にプログラムを引用し（印刷したものでも可）、１行１行どんな作業を行っているのかを説明すること。また、自分で新たな変数を宣言した場合はその変数が何を表しているのかも説明すること。

4.　　実験結果

それぞれのプログラムを実行して表示された結果を xwd を用いて取り込んで変換した PS ファイルを印刷し、レポートに添付すること。

5.　　考察

○ 結果について考察しなさい。

○その他、実験中に出た疑問点などを問題として設定し考察しなさい。

＜レポートに関する注意事項＞

- **プログラムが動き、正しい結果が得られて初めて合格の基準に達する。そのためレ　　ポートではしっかり動いているということをアピールするように書くこと。**
- **途中経緯や、失敗例等は貴重な成果となるので積極的にレポートに取り入れること。**
- **レポートは、図や表を積極的に活用し、読む人間に理解してもらうということを念頭に、わかりやすく書くこと。**
- **考察は、感想文や体験文ではない。自分で問題を設定し、考察する能力を身につけること。**
- **単にプログラムリストや結果を綴じただけのものや、項目に欠けがあるものはレポートとして認めがたいので受け取らない。**
- **他人のレポートや、プログラムの丸写しが確認された場合、いずれの側も減**

点とする。

- 表紙だけでの提出は認めない。プログラムの完成が間に合わなくても最低限書けるところがあるはずなので、それらを書いた上で提出時に相談すること。
- 時間割の都合上、課題を消化するためには実験時間だけでは不足するのが普通である。空き時間などを利用して課題には十分時間をかけること。基本的に CAD 室は空いているかぎり自由に利用して良い。

＜CAD 室の鍵の貸し出しに関する注意事項＞

- 鍵の貸し出しを新 2 号館 6 階の情報システム実験室で行うので、積極的に空いている時間を利用し、借りにくること。
- 貸し出しの際には名前、学籍番号を貸し出し簿に記入すること。
- CAD 室利用後はすみやかに鍵を返却すること。他の授業などでやむを得ない場合でも、当日中には必ず返却すること。
- 鍵の又貸しは決してしないこと。

2.2　提出場所・期限

提出場所：新２号館６階　一番奥の「情報システム実験室 (1)」「情報システム実験室(2)」

TA（ティーチング・アシスタント）の学生　田尻､呼

提出期限：実験終了後２週間以内

2.3　再レポートについて

レポートに不備があった場合、提出したその場、あるいは掲示で再レポートとする。合否の結果は提出後 1 週間以内に大島教官室に張り出される。電子メールでも連絡する。再提出の期限は再レポート受け取り後 2 週間後とする。

2.4　質問等について

大島教官、TA の学生（田尻、呼）まで、直接問い合わせるか、下記までメールすること。

新２号館６階　情報システム実験室（一番奥の部屋）
教授　大島 正毅　oshima@kdenshi.tosho-u.ac.jp
ＴＡ　田尻 大地　daichi@kdenshi.tosho-u.ac.jp
呼 _　hu@kdenshi.tosho-u.ac.jp

2.5 アンケートについて

今後の実験のためにアンケートを行っているので、別紙のアンケート用紙に返答し、レポートとあわせて提出すること。

◆◆ 参考書籍

本実験に於いては以下の書籍を参考にするとより理解が深まる。ただし、レポートに明らかにこれら書籍からそのままコピーしたと思われるものが発見された場合、再レポートととなるので注意すること。

- プログラミング言語C　B．W．カーニハン／D．M．リッチー著　共立出版
- C言語によるはじめてのアルゴリズム入門　河西朝雄著　技術評論社
- ＵＮＩＸプログラミング環境　B．W．カーニハン／R．パイク著　アスキー出版局
- X-window Ver.11 プログラミング／木下凌一・林秀幸著 日刊工業新聞社
- MAKE の達人　C.トンド／A.ネイサンソン／E.ヤント著 トッパン

3　UNIX の使い方

3.1　　login,logout について

UNIX システムを使用するには、ユーザはログイン (login)という操作を行わなければならない。ログインの方法は、自分のユーザ名とパスワードを打ちこむ。
同様に作業を終了する際にはログアウト(logout)という操作を行わなければならない(ログアウトを行わないと他人に勝手に使用される危険がある)。

● login について
c** login: という状態でユーザ名 ex12-**を
打ち込み、次にパスワードが要求されるので指定されたパスワードを打ち込む。
うまくログインできるとコマンドプロンプト%が表示されるので、
startx
と打ち込むとウィンドウシステムが立ち上がる。

● logout について
ログアウトするにはウィンドウシステムを終了させる操作が必要である。
方法はウィンドウシステム上で３つあるマウスのボタンの左ボタンを押してメニューを
出し、そこの下の方にある「EXIT」というところまでドラッグして離す。
すると、ウィンドウシステムが終了し、コマンドプロンプト%が表示されるので、
exit
と打ち込むと UNIX のシステムを終了できる。
＜注意＞ログアウトしている間は、キーボードに触れないこと。

● パスワードの変更
今回実験で用いる UNIX のシステムにおいては、パスワードを入れることでユーザであることを判定するので、これを変更しないと他人に悪用されるおそれがある。そのため以下の要領でパスワードの変更を行うこと。新しいパスワードは６文字以上で８文字程度にし、なるべくアルファベットだけでなく数字や記号なども入れると効果的である。
注意：変更するパスワードについては他人にはわかりにくく、かつ自分が忘れにくいものにすること。
変更の仕方：
ウィンドウシステム上で３つあるマウスのボタンの真ん中ボタンを押して「kterm」というターミナル（コマンドをいれたりする作業をするためのウィンドウ）を開き
「yppasswd」と打ちます。その後「Old password:」と聞いてくるので最初に変更前の

パスワードを入れる、そしてその後「New password:」となるので新しいパスワードを入力する。それを打ち終わるともう１度新しく入れたパスワードを入力するように要求されるのでもう１度入力する。

変更の確認：
変更の確認はいったん logout して、もう１回 login できることを確認すること。

3.2 基本的なコマンド

ls　自分が現在いるディレクトリ（カレントディレクトリ）の内容（ファイルやディレクトリ）を表示させる。

pwd　カレントディレクトリ名を表示させる。

cd　別のディレクトリへ移動させる。
使用方：cd ディレクトリ名
ただし、ディレクトリ名を入れず cd だけだと
自分のホームディレクトリに戻る。

mkdir　ディレクトリの新規作成。
使用法：mkdir 新規ディレクトリ名

cat,more　（テキスト）ファイルの内容を表示させる。
more の場合は、一画面ずつ表示させる。
使用法：cat もしくは more ファイル名

cp　ファイルをコピーする。
使用法：cp 元ファイル名 新ファイル名（もしくはそのファイルを置きたいディレクトリの場所（パス））

mv　ファイルの移動及びファイル名の変更。
使用法：mv 元ファイル名 新ファイル名（もしくは移動先ディレクトリ名）

rm　ファイルの削除。
使用法：rm ファイル名

rmdir　空ディレクトリの削除。
使用法：rmdir 消したいディレクトリ名

ln　リンクを張る。

使用法：ln –s リンク元のディレクトリ名 リンク先のディレクトリ名

man　マニュアルコマンド。コマンドのマニュアルを表示する。
使用法：man コマンド名

3.3　エディタの使い方

テキストを作成したり、編集修正をしたり、ファイルに入出力するためのソフトウェアをエディタという。ここでは代表的なエディタである emacs についての使い方を載せる。

● Emacs の起動

emacs ファイル名

● カーソルの移動

１語上に移動　CTRL+p
↑
１語左に移動　CTRL+b ← ◎ →１語右に移動　CTRL+f
↓
１語下に移動　CTRL+n

行頭に移動	CTRL+a	行末に移動	CTRL+e
文頭に移動	ESC+<	文末に移動	ESC+>
１画面分前に移動	ESC v	１画面後ろに移動	CTRL+v
指定した行に移動	CTRL+o 行数		
指定した文字列に移動	CTRL+S ESC 文字列		
文字列検索	CTRL+s 文字列		

● 削除

１文字削除	CTRL+d	１行削除	CTRL+k

"CTRL+k"によって最後に削除されたテキストを取り込む　CTRL+y
直前の状態に戻す(undo)　CTRL+_

● 改行

ポイントの後ろを何行か改行　ESC 数字
ポイントの後ろを１行改行　CTRL+m

● 置き換え

文字列置換　ESC %　置換対象となる語 置換する語
置き換えて次へ進む　SPACE
置き換えるが移動しない　,

置き換えずにスキップ	DEL
残りをすべて置き換える	!
直前に置き換えた位置に戻る	^
再帰編集に入る(ESC-CTRL-c で抜ける)	CTRL-r
終了	ESC

● ウィンドウ

ウィンドウを全画面に	CTRL+x 1
ウィンドウ分割	CTRL+x 2

● バッファ

バッファ切り替え	CTRL+x b バッファ名
バッファを閉じる	CTRL+x k バッファ名

● ファイル操作

ファイル挿入	CTRL+x CTRL+i ファイル名
別ファイル読み込み	CTRL+x CTRL+f ファイル名
ファイルのセーブ	CTRL+x CTRL+s
別ファイル書き出し	CTRL+x CTRL+w ファイル名

● Emacs の終了

CTRL+x CTRL+c

3.4　コンパイルと実行

作成したCのプログラム名を***.c というように最後に「.c」と付くようにする(***には自由に名前を付けてよい)。作成したプログラムを実際に実行（使えるように）させるためには「コンパイル」という作業を行う必要がある。
コンパイルには、いくつかの方法がある。もっとも基本的な方法は、「gcc ***.c –o ***」と打つことである。
エラーがでてくれば、そのエラーメッセージを参考にして修正し
エラーが出てこなくなったら、「***（必要ならファイル名も）」と打つことでプログラムを実行できる。

ここで、コンパイルをする際には、エラーが出なかったがうまくプログラムが動かない、といった場合にはデバッガ（下参照）を用いることによって解消することができる。
今回は、画像表示にＵＮＩＸのウィンドウシステムである「X-Window」を使用するため、複数のライブラリーを読み込む必要がある。そのため、Makefile を用いたコンパイルを行う。Makefile の基本となるものは用意してあるので、その１行目に書かれているファイル名の情報をコンパイルしたいファイルのものに変更して、「make」と打ち込むことでコンパイルすることができる。
→　参考文献　MAKE の達人　C.トンド／A.ネイサンソン／E.ヤント著 トッパン

● デバッガ

少し複雑なプログラムを作る場合、予期せざる誤りを取り除く操作（デバッグ）を完成するまで何度も繰り返すことになる。文法上の誤りは gcc（c コンパイラ）が指摘してくれるが、文法上は正しくても思い違いなどから求める結果が得られないことがよくある。このようなとき、途中の経過（どの段階で変数の値がいくらになっているかなど）をプリントさせてみたりすると（また、もし可能ならプログラムの実行を途中で止めてみる）プログラムの動きを確認でき早くデバッグできる。そのためにプログラムに本来は必要ない printf 文（時として入力文も）などを挿入することも行われる。これも役立つが、もっと便利なやり方がある。それがデバッガである。

● デバッガの使い方

コンパイルするとき gcc に代えて gcc -g とする（これをデバッガオプション付のコンパイルという）。デバッガオプション付のコンパイルを行うと多少実行速度が遅くなるが、できあがった実行プログラム（a.out）をそのまま通常通り実行することもできる。

デバッグモードの実行は次のように行う。
コマンドが実行できる状態（コマンド行）で dbx a.out と入力する。実行プログラムが読み込まれデバッグモードとなる。この状態で種々の dbx のコマンドが入力できるようになる。

dbx コマンドの主なものは：

help	dbx コマンドとしてどんなものがあるか教えてくれる。help に続けて dbx コマンド名を入れれば dbx コマンドの使い方を教えてくれる。
run	プログラムを開始する。どこか途中で止まっているときはご破算でやりなおしとなる。
list	ソースプログラムを番号付で表示する（この番号をよく使う。なお、番号付のリストを手元に置きたいときは cat -n コマンドを利用できる）。
print	指定した変数の値を表示する。
stop at	指定した番号に制御が移るとき実行直前で止まる。
stop in	指定した関数に制御が移るとき実行直前で止まる。
step	止まっているとき、１ステップだけ次に進む。
next	step と似ているが、step は実質的な１ステップ次に進む（例えば関数呼び出しの直前で止まっているとき関数の中に入って止まる）が、next はソースプログラム上で次へ行く（関数の中へ入り込まないで、形式的にその行を実行する）。
cont	止めたプログラムを継続する（従って次に stop at や stop in に出会うまでどんどん進む）。
dump	使っている変数の一覧を表示する。
quit	デバッガを終了する。

デバッガを使いこなすとプログラムのデバッグが早くできるし、プログラムの働きを深く理解できる。なお、システムに備えられている関数はデバッグモードでコンパイルされていないので、中に入って見ることができない。また gcc コンパイラにおいては、main プログラムが意図するように表示されないことがあるが、その場合、main プログラムの名前を MAIN に変え、形式的に設けた main プログラムから呼び出すようにすればよい。（注意、デバッグモードでの実行はスピードを除きほぼ完全に通常の実行と同じと考えてよいが、ごくまれに結果がことなることがある（似たような事情は実行速度を上げるためのオプティマイゼーション付コンパイルでも存在する）。）

3.5　電子メールの概要と使い方

電子メールとはコンピュータ上で電子の郵便のやりとりをすることで使い方は、「mail　宛先のユーザ名（アドレス）」で出すことができる。届いたメールを読む場合には、「mail」とだけ打つことにより読むことができる。

実験や UNIX 全般についての質問は電子メールでも受け付ける。積極的に利用することが好ましい。質問先のアドレスは 3.4 を参考にすること。

現在 cad システムにおいて漢字の入力方法は整備されていない。情報処理センターから漢字を含むメールのやり取りができる。

また、上記いずれのシステムともインターネット接続されている(今日の常識)ので、簡単にデータの転送ができる。

1999 年 4 月 16 日　　1999 初版
1999 年　　第 2 版

編集者　田尻 大地、
呼 _、上原 将文

監　修 大島　正毅