本稿は Linux Japan 誌 2002 年 03 月号に掲載された記事に補筆修正したものです.

Expect

Expect[1]W[3] は Tcl の拡張のなかでも最も古くから開発され利用されてきました. 生みの親 J. K. Ousterhout 氏の手による Tcl のバイブル “Tcl & Tk ツールキット” [2] にも拡張の筆頭にあげられています. 開発者は NIST （National Institute of Standards and Technology） Don Libes[3]W[3] 氏で 1990 代 初頭より Tcl 上での開発が始まったようです. この拡張言語は Libes 氏自らの著書 “Exploring Expect”[4]W[3] の副題にもあるように，対話的なプログラムの制御を自動化するためのツールキットです. 対話的なプログラムとは，ftp，telnet，ftp，fsck などをさしています. これらのプログラムはユーザーにパスワードなどの入力や確認のためのキー入力などを求めてきます. 一般にはそのメッセージに対してその都度キー入力を行なわなければなりません，もちろんこれでは不便ですから，スクリプトを組んでメッセージのやりとりを自動化したいと思うわけです. ところが，シェルのリダイレクトやパイプなどの組み込み機能だけではプログラムが対話モードになった場合にプログラムからのメッセージを取り込むんで制御することはできません. そこで Expect の登場となるわけです.

Expect なしでどうにか

上記の事情を確かめるためにいくつか実験をしてみましょう. まず ftp や telnet（これが禁止されているなら安全な sftp や slogin ）とパイプでつないだらどうなるかですが,

```
$ echo "***" |sftp rhost ↵
***
Enter passphrase...  'user@localhost':
sftp> Connection closed
```

のように echo を使ってパスワードを sftp に送りリモートホストにアクセスはできるのですが，echo の終了とともに sftp も終了してしまいます. ちょっと工夫して,

```
$ (echo "***";cat)|sftp rhost ↵
***
Enter passphrase ...  'user@localhost':
get xli-1.16.tar.gz ↵
stp> xli-1.16.tar.gz:  ..................
296876 bytes received ...  166.29 K/s
quit ↵
sftp> Connection closed
↵
$
```

と ftp のセッションを継続することは可能です. しかし，一連のコマンドをファイル（仮に ftp.cmd としましょう）に書いておいて，cat を使って送り込もうとすると,

```
$ (echo "***"; sleep 5; cat ftp.cmd)
  |sftp rhost ↵
```

のように，途中に sleep を入れて同期？と取らないといけません.

そういうようなわけで，通信手続きをが固定されていれば，なんとかなります. しかし，相手の状態に応じて手続きを変化させるなどという芸当は不可能です.

もっとも ftp は $HOME/.netrc に host 名，login 名，password を記述しておくと，password 認証が自動的に行われますから,

```
$ ftp rhost <ftp.cmd ↵
```

というように実行できますし，sftp はオプション `-f cmdfile` によって，コマンドを記述したファイルを指定することが可能ですから,

```
$ echo "****" |sftp -f ftp.cmd rhost ↵
```

のように実行できてしまいます.

Expect を使うと

Expect を用いればもっと簡単にスクリプトが組めます. また，繰り返しますが，対話型プログラムのメッセージに応じて，手続きを選択するような複雑なスクリプトを組むこともできるようになります.

インストール

近頃ではバイナリ配布が安定してきて，あまりインストールの話をしなくても済むのですが，Expect の場合には礎となっている Tcl/Tk との依存関係が無視できませんので調べる羽目になりました. まず筆者愛用の Plamo-2.x では Tcl/Tk のバージョンが 8.0.2，Expect は 5.28 です. Vine-2.1.5 では Tcl/Tk が 8.0.5，Expect は 5.28 です. Tcl/Tk のこのバージョンは安定していますが，その上でコンパイル可能な Expect は 5.30 までのようです. これ以降 Tcl/Tk は Unicode を採用した国際化が進み，Expect もそれに応じて機能的にはあまり変化してませんが，書き換えが必要となったようです. RPM パッケージを rpmfind.net[5]W[3] で探すと，Mandrake Linux が 8.3.3 などというバージョンを付けてますが，これは Tcl/Tk のバージョンをそのまま付けた間違いでしょう. 日付から察すると多分 5.32 と思います. Tcl/Tk-8.3 と Expect-5.34 という組み合わせが最新の状況で，パッケージ化されてないようですね. ソースファイルからコンパイルし直す場合には（Tcl/Tk のライブラリと include ファイルが要求されます），組み合わせに注意してください.

主なコマンド

この言語の特徴を表しているのは **expect** と **send** です. uucp の時代の名残だそうですが，expect は相手からのメッセージを待ち，予期された文字列が届いたら，それに応ずるコマンドです. send は相手にメッセージを送るコマンドです. さて相手（プロセス）を起動しなければなりません，**spawn** がその任にあたります. もう 1 つ重要なコマンドは **interact** です. このコマンドが発効された後には，基本的には交信メッセージが透過になり，パターンを指定した文字列については変換されて送られるようになります. したがって何もパターンを指定しなければ，spawn されたプロセスと直接交信しているようにみえます. この 4 つのコマンドが全体の流れを決定していますから，それだけは最低限覚えましょう.

細かいことは簡単な例を読みながら紹介していきます. まずは，典型的な例として，リモート先に “`slogin`” して “`mnews -m`” によりメールを読めるようにするスクリプトです. 名前を仮に smnews.exp としましょう.

```
#!/usr/bin/expect --

set f [open "/home/matuda/.password" r]
gets $f passwd

log_user 0
spawn slogin -2 $argv
expect "password: " {send "$passwd\r"}
send "inc\r"
send "mnews -m -n MH\r"
interact  "^]" {return}
system "clear"
exit
```

いきなり Tcl のコマンドを使ってしまいました．Expect は Tcl の拡張言語ですから Tcl の基本コマンドは当然ながら全て使うことができます．この例ではパスワードを変数 `passwd` に与えています．`log_user 0` は余分なメッセージを抑制するためのコマンドです．spawn で `slogin -2` に smnews.exp のコマンドライン引数の内容（= リモートホスト名）`$argv` を渡して起動します．パスワードを要求するプロンプト `password:` を受けたならば，パスワード `$passwd` を slogin に送ります．改行コード`\r` を忘れないでください． 多分ログインできますので，コマンド `inc` と `mnews -m` をリモート先のシェルに送ります．あとは，interact で透過モードとなり，通常通りメールを読むことになります．ただし，キャラクター“`^]`”（制御文字 `0x1d` であり，文字 `^` と `]` の並びではありません．Emacs では `C-q` の後に `C-]` をキー入力します）がユーザー側からキー入力されると interact を抜けます．`system` でシェルに clear（画面の消去）を実行させます．

mnews -m を起動するまでの実行の様子は，以下のようになります

```
cj3027512-a:~$ smnews.exp namio ↵

Last login: Fri Jan  4 12:26:44 2002
Linux 2.2.16.
inc
namio:~$ inc
inc: no mail to incorporate
namio:~$
```

空行が目立ちますが，`log_user 1` に設定した場合と比較してください．

```
cj3027512-a:~$ ./smnews.exp namio ↵
spawn slogin -2 namio
matuda@namio.....jp's password:
Last login: Fri Jan  4 12:26:44 2002
Linux 2.2.16.
inc
namio:~$ inc
inc: no mail to incorporate
namio:~$
```

“`mnews -m -n MH`” を活かした smnews.exp では，これらが一瞬表示された後 mnews の画面にかわります．

ファイルからパスワードを得るために Tcl コマンドを使いました．Expect で定義されたコマンドでパスワードをその場入力する方法を用いてみましょう．

```
#!/usr/bin/expect --

stty -echo
send_user "Password?  "
expect_user -re "(.*)\n"
set passwd $expect_out(1,string)
send_user "\n"

puts "You psssword is $passwd"
```

まず `stty -echo` で，キー入力した文字（この場合にはパスワードなので大変気を使います）がエコーバックされないように端末を設定します．`send_user` でユーザー側にパスワード要求のメッセージをだし，`expect_user` でユーザーの入力に一致を試みます．ここでは，オプション `-re` により正規表現であることが明示されています（一致の成否はシェルのワイルドカードの展開 Glob がデフォルトです．これを明示するには `-gl` をつけます）． 正規表現の内容はご存じと思いますが，`.` が任意の文字で `*` がその繰り返し，`( )` はグループ化です．`$expect_out(0,string)` が一致文字列全体 (つまり`\n` を含みます)，`$expect_out(1,string)` がグループ化された最初の文字列の内容となります．それを passwd に set します．最後のコマンドは Tcl の基本コマンド puts でパスワードが本当に入力されたかどうか確かめるためのものです．もちろん smnews.exp では必要ありません，いや，パスワードの表示などあってはならないものでしょう．

人がしゃべるように

他に類をみない面白い機能を紹介します．`-h` をつけた場合の send です．Humanly の略で，文字を人がキー入力しているように表示します（カラオケの歌詞の感じ）．表示の間隔は

```
set send_human{p1 p2 p3 p4 p5}
```

なるコマンドの，5 つのパラメータで設定します．最初の 2 つは文字を表示する平均の時間で第 2 番目は単語の終了時に用いられます．3 番目は揺らぎを設定していて値が小さい程間隔が揺らぎます．最後の 2 つは，時間間隔の最小値と最大値です．実際に試せばすぐに理解できます．次のっようなスクリプトを coffe.exp とでもして作成してください．

```
#!/usr/bin/expect --

set send_human {.2 .4 1 0.1 1}
send -h "コーヒーでもどう？\n"
```

実行属性を与えて動かして観ましょう．

```
$ chmod +x coffe.exp ↵
$ ./coffee.exp ↵
コーヒーで...
```

文字がポツリポツリと表示されたでしょうか？

`-s` オプションをつけると一定の間隔でゆっくりと表示をします．設定は

```
set send_slow {p1 p2}
```

と 2 つのパラメータを与えます．第 1 番目は同時に送る文字数，第 2 番目はその間隔を秒単位で指定します．次のようなスクリプト（仮に slowread.exp としましょう）が考えられます．

```
#!/usr/bin/expect --

set fid [open "$argv" r]
set send_slow {1 .08}
while (1) {
   gets $fid str
    if { [eof $fid] } then exit
   send_user -s "$str\n"
}
```

最初からゆっくり読みたいファイル（filename）を指定して動かす

と，less よりも読みやすいかもしれません．

```
$ ./slowread.exp filename ↵
```

Expect に付属のツール

Expekt を make install すると，5.30 では expect と expectk（expect の拡張を理解する Tk）の他に，以下のスクリプトがインストールされます．

```
timed-run, timed-read, ftp-rfc, autopasswd,
lpunlock, weather,
passmass, kibitz, rlogin-cwd, xpstat, tkpasswd,
dislocate,
tknewsbiff, unbuffer, mkpasswd, cryptdir,
decryptdir, autoexpect,
xkibitz
```

これらスクリプトの中身をじっくり観察すれば，おのずと Expect の Expert になれるはず，なんちって (^^; 駄洒落はさておき，その中で発想が興味深いあるいは面白いものを紹介して今回の話を締めくくりましょう．

xkibitz　1 つのファイルを複数のメンバーが一斉に編集をするなどということはあまりないと思いますが，もしそのような事態が生じたらどうしましょう？ kibitz または xkibitz がそのような共同作業を可能にします．使いやすい xkibitz の方を簡単に説明します．最初に起動された xkibitz がマスターとなります．端末上には

```
Escape sequence is ^]
```

が表示されて，またシェルに戻りますから xkibitz が動いていることが判り難いです．そこでメッセージの言うとおり C-] を押してみてください．xkibitz のコマンドモードに入ります．ここでさらに ? を押せば，コマンドが表示されます．

```
for help enter:  ?  or h or help
xkibitz> ?  ↵

Commands          Meaning
--------          -------
return            return to program
=                 list
+ <display>       add
- <tag>           drop
where <display> is an X display name such
as nist.gov or nist.gov:0.0
and <tag> is a tag from the = command.
+ and - require whitespace before argument.
return command must be spelled out
("r", "e", "t", ...).
xkibitz>
```

説明の通り “+” を用いて，X サーバー（実際にはその上の xterm）をスレーブユーザーとして追加できます．もちろん同じ X サーバーを使うことも可能で（意味ありませんが） unix:0 あるいは単に :0 を追加してみましょう．

```
xkibitz> + :0 ↵
xkibitz>
```

すると，マスターと同じ大きさの xterm が起動します（スクリプト中 1 箇所 kterm に書き直せば kterm が起動します）．もちろんリモートホストの X サーバーを指定すれば（もちろんアクセス権限が必要です），リモート先に同じ大きさの *term が現れます．コマンドモードから戻るには 文字列 “return” を打ち込みます．で，何ができるかというと，その上でのキー入力が完全に同じになるのです．どちらから入力あるいは消去も可能です．したがって，mule -nw などを動かせば，共同開発者が 1 つのソースを同時に編集することも可能になります（図 1）．シェルの上で誰かが exit を入力すると xkibitz は終了してしまい，スレーブの *term は消えてしまいます．

xkibitz の man によれば，初心者が上級者に添削をお願いする用途にも使えるだろうとあります．もちろん，マスター側が初心者（つまり初心者の権限で実行されます）でないといけませんね．

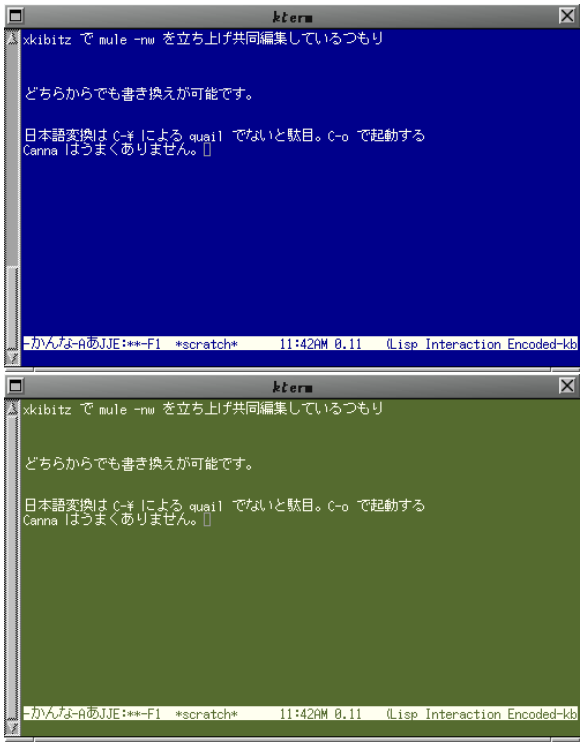

図 1　kterm を起動するように書き換えた xkibitz を動かし，mule -nw によって共同編集を模擬している様子

mkpasswd　多数のユーザーのアカウントを一斉に登録するには，パスワードを準備しなければなりません．mkpasswd はランダムなパスワードを生成してくれます．全文字数，大文字の数，数字の数をオプション指定できますから，例えば全 8 文字で大文字 1，数字 2 を含むパスワードを生成してみると以下のようになります．

```
$ mkpasswd -l 8 -C 1 -d 2 ↵
qN8hdde1
```

すべての新規ユーザーに対する仮パスワードを mkpasswd で作成し，newusers を用いて一期に登録をすませられるというわけです．

tkpasswd　mkpasswd では，ユーザー名を与えると新しいパスワードに変更しようとします．しかし，/bin/paswwd が古いパスワードを要求する場合，mkpasswd はそれを想定してませんから失敗します．tkpasswd は Expectk で作成されたツールで，その辺りの事情が考慮されており，GUI でパスワードを変更します．tkpasswd の起動直後の画面を図 2 に示します．ユーザーを選択して，自分でパスワードを入力するか，自動生成してもらうかして “set” ボタン

を押すと，古いパスワードを尋ねるダイアログが現れます．そこに正しく古いパスワードが入力されれば新しいパスワードに変更されます．

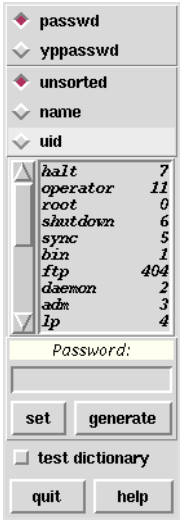

図 2　tkpasswd の起動画面

passmass　NIS を使わずにパスワードを共通管理するのはかなり大変です．考えるのも嫌ですが，管理者はパスワードを変更したホストの /etc/shadow を他のホストにもコピーしなければなりません．パスワードの変更はユーザーの仕事ですから，全部のホストのパスワードを一斉に変えてもらいましょう．Expect ならばホストのリストがあれば，そのようなことが可能であることは明らかです．要するに，passwd コマンドによる変更を全てのホストにログインして実行すればいいだけのことですから．passmass は rlogin を用いていますが，これを slogin に変更することは容易でしょう．

参考文献

[1] Expect のホームページ．W3
`http://expect.nist.gov/`

[2] J. K. Ousterhout 著，中西 芳幸，石曽根 信 訳：“Tcl & Tk ツールキット”（ソフトバンク，1995）．

[3] Don Libes 氏の自己紹介ページ．W3
`http://www.mel.nist.gov/msidstaff/libes/`

[4] Don Libes, “Exploring Expect – A Tcl-based Toolkit for Automating Interactive Program” (O’Reilly, 1994). W3

[5] RPM の検索サイト rpmfind.net. W3
`http://rpmfind.net/`