Linux Japan

Full Screen Window Manager

Screenを使う

井口剛
zinnia@jan.ne.jp

■前書き

本稿では、Screenとはどのようなソフトで、どのようなことができるのかを紹介します。ページ数の都合上そのすべては扱いきれませんが、自由な発想でScreenを活用できるために必要な道具は一通り解説したいと思っています。

■Screenの概要

Screenとは、(ktermなどの)1つの端末の中に仮想的な端末「ウィンドウ」を作り、操作を行うことができるウィンドウマネージャです。表1のような特徴があります。Screenはウィンドウの集まりを「セッション」として管理しています。セッションと端末は柔軟に結び付けたり、切り離したりすることができます。例えば

- 端末Aで動かしていたScreenを切り離して、また端末A接続する
- 端末Aで動かしていたScreenを切り離し、別の端末Bに再接続する
- 端末Aと端末Bで同時に1つのセッションを操作する

といったことが可能です。セッションが端末から切り離されても、その中のプログラムは動き続けています。Screenで使う用語を表2にまとめました。

■情報源

●Screenそのものに関する情報

何よりもまず配布ファイルを入手しましょう。Screenのマスターサイト(記事末のResource[1]を参照)を参考に、手近のミラーサイトから入手してください*1。配布ファイルの中には、ソースはもちろん、多くの有益なドキュメントが含まれています。

記事執筆現在の最新版は3.9.10です(他の多くのソフトの配布ファイルと同様に)配布ファイル内のNEWSというファイルを読むことで、以前のバージョンからの変更点を手軽に知ることができます。新しい機能が追加されたときは簡単な使い方も説明してありますので、バージョンアップのたびにチェックするようにしましょう。

●Screenの使い方に関する情報

・ヘルプ

Screenが動いてる端末で「^A?」(Ctrl＋a＋?。ここでは、デフォルトの表記に従います)とすることでヘルプを見ることができます(画面1、画面2)。上にある「Command key: ^A Literal ^A: a」で、コマンドキーが^Aで、^Aそのものを入力したいときは^Aaと入力すればよいことが分かります。ヘルプの内容は「コマンド名とキーの対応表」です。「time t」とありますか

【表1】Screenの特徴

・複数のウィンドウを1つの端末で管理できる
・ウィンドウは2ストロークで簡単に切り替え可能
・ウィンドウは2ストロークで簡単に増やせる
・ウィンドウ間でコピー&ペーストを行える
・ハードコピー、ログをとることができる

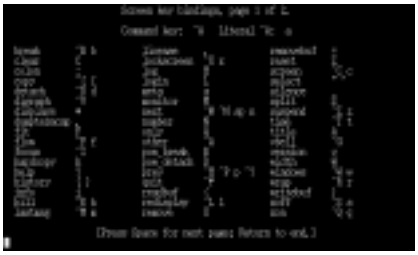
【表2】用語の説明

用語	意味
物理端末	物理端末は、Screenが動くことのできる(本物の)端末です。kterm、eterm、rxvt、mltermなどなどです。
ウィンドウ	Screenは端末の中でウィンドウと呼ばれる仮想端末を作ります。ウィンドウにはそれぞれ番号が付いており、名前を付けられます。
コマンド	Screenに対する指示をコマンドといいます。コマンドは次に述べる「コマンドキー」＋「別のキー」のシーケンスに割り当てることができます。また、コマンド名を直接指定してScreenに指示することもできます。コマンドの詳細は後述します。
コマンドキー	Screenのコマンドを発行するために最初に入力する文字です。デフォルトでは「^A」となっています。escapeコマンドで指定します。なお、Screenのコマンドでは「^」＋「文字」という表記をCTRL＋「文字」と理解します。本記事でもこの表記に従っています。
リテラル	「コマンドキー」で指定された文字そのものを入力したいときに使う文字を指定します。デフォルトでは「a」になっております。例えば、コマンドキーが^Aで、リテラルがaのときに、^Aそのものを入力したければ^Aaとすればよいことになります。リテラルもescapeコマンドで指定します。
レジスタ	コピーコマンドなどで取り込んだ文字列を保存しておくための領域をレジスタといいます。レジスタにはキー(名前)が付いており、複数のレジスタを管理することができます。
メッセージライン	Screenに対する指示の応答のための行をメッセージラインといいます。メッセージラインは、ステータスライン(注)、端末の中(最下行)などに表示させることができます。メッセージラインを常に表示させておくこともできますが、デフォルトでは必要に応じて表示され、一定時間後に消えます。
セッション	1つのScreenが管理しているウィンドウの集まりをセッションと呼びます。セッションは端末に結び付いていますが(アタッチ)、これを切り離したり(デタッチ)、再接続させたり(レジューム)することができます。
ディスプレイ	物理端末です。Screenのドキュメントでは、特にScreenのセッションに結び付けられた端末のことをディスプレイと呼ぶようです。
アタッチ	セッションが端末に結び付けること、またはその状態です。
デタッチ	セッションを端末から切離すこと、またはその状態です。
レジューム	デタッチされたセッションを端末に再接続させることをレジュームといいます。
アクティブ	ウィンドウが操作可能な状態にあることをいいます。

(注)端末に備わる各種情報表示のために用意された行。ktermでは、CTRL+2で出てくる「VT Optionsメニュー」で、「Enable Status Line」をチェックするか、リソースを設定することで使えるようになります。

＊1 最新の動きなどは、freshmeat([2])などから仕入れたほうがいいかもしれません。

【画面1】ヘルプ画面1



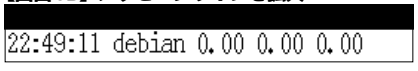
【画面2】ヘルプ画面2

【実行例1】

```
$ tar zxvfp screen-3.9.10.tar.gz
$ cd screen-3.9.10
$ ./configure
$ make
$ su
# make install
```

【画面3A】メッセージライン（ステータスラインに表示）

【画面3B】メッセージラインを拡大



【リスト1】

```
termcap entry (./terminfo/screencap) should be installed manually.
You may also want to install ./etc/etcscreenrc in /usr/local/etc/screenrc
```

ら、^Atと入力するとtimeというコマンドが実行されることが分かります。timeコマンドはinfoやマニュアルを見れば分かる通り、時刻、ホスト名、load averageを表示するコマンドです。

コマンドは、だいたいの動作が想像つくような名前が付いています(少なくともマニュアルを引くための手がかりにはなります)から、慣れないうちはヘルプを見ることで「自分のやりたいことはどういうキー操作でできるのか」ということを知ることができます。逆に「あのキー操作はどういうコマンド名になっているのか」ということを知るためにヘルプを見る、といった使い方もできるわけです。

・info

Screenのinfoは素晴らしく、Screenとはいかなるもので、どのようなことができて、それをどのように使うのかということが網羅されています。つまりここで私が書こうとしていることは、ほとんどがinfoにも書かれているということです。1度は目を通しておいて、どんなことが書かれているかを把握しておくことをお勧めします。

●マニュアル

infoほど詳細ではありませんが、コマンドライン引数やScreenのコマンドに関する一通りの説明があります。分量が少ないのでまずこちらをあたってから、必要に応じてinfoを調べてみるといいかもしれません。またJM Projectの成果により、和訳されたマニュアルを入手できます（[3]）。

■Screenのインストール

ほとんどのディストリビューションではパッケージが用意されているでしょうから、そちらを利用するといいでしょう。ここではソースからのインストールのみを解説します。インストールはDebian GNU/Linux 2.2上で行いました。

ソースからのインストールは、他のGNUソフトウェアと同様に、configureスクリプトで設定を行った後にmakeを行い、rootになってmake installでOKです（実行例1）。make installの最後にリスト1のようなメッセージが表示されます。必要に応じてtermcapエントリを追加してください*2。

設定ファイル「screenrc」については、指定された場所にコピーしておきます。/usr/local/etc/screenrcには共通の設定を記述し、$HOME/.screenrcに各自の設定を置くという使い方をするとよいでしょう。または、$HOME/.screenrcに./etc/etcscreenrcの中身を入れて、それを編集するというやり方もあると思います。利用者の運営方針に合った方法を使ってください*3。

本稿では、以後.screenrcと書いた場合はホームディレクトリの.screenrcを指すものとします。

■設定しておきたいこと

設定ファイル内「.screenrc」でとりあえず指定しておきたいものを以下にまとめておきました。

●コマンドキー、リテラル指定

Screenのコマンドを呼び出すために最初に入力する文字です。「用語の説明」でも述べた通り、デフォルトでは、コマンドキーが^A、リテラルがaとなっております。

コマンドキーとリテラルはescapeというコマンドで指定することができます。例えば

```
escape ^Tt
```

とすると、コマンドキーを^T、リテラルをtとすることができます。このとき、「^T」は「^T」という文字、ではなく「^」と「T」の2文字を

そのまま入力します*4。

コマンドキーのデフォルトの^Aは、Emacs系のエディタやシェルなどで「行頭に移動」するために割と頻繁に使う人が多いと思います。「Screen内で『行頭に移動』するために、わざわざ^Aaと入力しなければならないのは面倒だ」という場合は、上記のコマンドで自分が普段あまり使っていないキーを割り当てるようにしてみてください*5。

なお本稿では、デフォルト設定の

```
escape ^Aa
```

をそのまま使うことにしますので、皆さんの環境に合うように適宜読み換えるようにしてください。

●メッセージラインの表示位置

Screenでは、端末のステイタスラインが使えればそこをメッセージラインとして使います(画面3A、画面3B)。ステイタスラインが使えない場合は端末の最下行を表示のために使用できます。

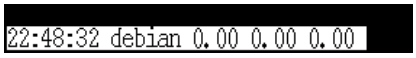
```
termcap kterm hs@
terminfo kterm hs@
```

という設定を.screenrcに書き込んでおくと、ステータスラインを使わずに、端末の最下行をメッセージラインとして利用可能になります(画面4A、画面4B)。また、リスト2のようにするとタイトルバーをステータスラインの代わりに使えます（画面5）*6。本稿の画面写真では、端末の最下行をメッセージラインとした設定をし

＊2　Debian GNU/Linuxではtermcapではなくterminfoを使っているため、この操作は必要ないようです。terminfoエントリを追加するためには、tic ./terminfo/screeninfo.srcとします。詳しくは./terminfo/READMEをご覧ください。　＊3 なお、ここでインストールされた設定ファイルは、いくつかのデフォルト動作を無効にする設定が入っています。そのため、infoにあるキーバインドとは必ずしも一致しません。　＊4　viだとCtrl+V Ctrl+T、emacsだとCtrl-Q Ctrl-Tとすることで「^T」という文字そのものを入力できます。が、Screenに対する指示のときはそのようなことをする必要がなく、単に「^」と「T」の2文字を入れればよいということです。　*5 私は^Tとしています。他には^]などが好まれているようです。

【画面4A】メッセージライン（端末の最下行に表示）

【画面4B】メッセージライン拡大



たときの表示を使用しています。

●漢字コード

Screenで日本語を正しく表示させるためにはtermcap/terminfoのKJエントリにエンコーディングを指定する必要があります。Screenではkanjiと、defkanjiというコマンドがこの設定をしてくれます（termcap/terminfoコマンドで設定することもできます）。kanjiコマンドはアクティブなウィンドウのエンコーディングを指定します。defkanjiは、新しくウィンドウを作成するときのデフォルトのエンコーディングを指定します。どちらのコマンドも、指定できるのはJIS、EUC、SJISのいずれかです。EUCに設定したいときは、

```
kanji euc
defkanji euc
```

というコマンドを.screenrcに書いておきましょう。なお、screen (-r,-x)でレジュームしたときに表示が崩れるといったときも、kanjiコマンドを送り直してから(^A:kanji euc)、再表示(^L)することで復帰させられます。

【リスト2】

```
termcap kterm 'hs:ts=\E]2;:fs=\007:ds=\E]0;screen\007'
terminfo kterm 'hs:ts=\E]2;:fs=\007:ds=\E]0;screen\007'
```

【画面5】メッセージライン（タイトルバーに表示）

■知っておきたいこと

●コマンドとキーバインド

Screenに対する指示はコマンドによって行います。例えば「次のウィンドウに切り替え」という指示は、「prev」というコマンドをScreenに送ることで実現されます。コマンドを送る方法には以下のような方法があります。

・.screenrcに列挙しておく

.screenrcに、Screenのコマンドを記述することで、Screenの起動時にそれらが順番に実行されます。

・コマンドキー入力による実行

Screenのコマンドは、「コマンドキー＋文字」に割り当てられます。例えば、prevコマンドはデフォルトで^A^Pと、^Apに割り当てられています。

キー割り当ての変更はbindコマンドによって行います。例えば

```
bind P prev
```

というコマンドを.screenrcに書いておくと、^APでも「前のウィンドウに切り替え」を実行できます。

・コマンド入力モード（^A:）に入って実行

^A:にはcolonというコマンドが割り当てられています。colonコマンドでは、Screenのコマンドをその場で実行させられます。例えば、^A:prevとすると、prevコマンドが実行され、前のウィンドウに切り替えることができます。

・screenの-Xオプションで実行

-Xコマンドラインオプションを与えてScreenを実行することで、Screenのセッションの外からでもコマンドを送ることができます。Screenの動いている端末とは別の端末のシェルから

```
$ screen -X prev
```

というコマンドを実行すると、Screenでprevコマンドを実行できます。この機能はバージョン3.9.9から追加されたものです。

●キーストロークとオプション

Screenの主なキーストロークを表3に示します。また、Screenのコマンドラインオプションのうち、比較的よく使うものを表4に示します。

【表3】主なキーストローク一覧（コマンドキー「^A」は省略）

利用キー	機能
:	コマンド実行（colon）
?ヘルプ（help）	
[または ^[または ESC	コピー（copy）
]または ^]	ペースト（paste）
a	コマンド文字そのものを出力
A	タイトル変更（title）
C	クリア（clear）
c または ^C	ウィンドウ作成（screen）
\	Screen終了（quit）
^A	直前にアクティブだったウィンドウに切り替え（other）
d または ^D	デタッチ（detach）
D	パワーデタッチ（pow_detach）
f または ^F	フロー制御 on／off／auto の変更（flow）
^G	ベルの変更（audible／visual）
i	ウィンドウの情報（info）
K	ウィンドウを閉じる（kill）
^L	再表示（redisplay）
m または ^M または Enter キー	直前のメッセージを表示（lastmsg）
p または ^P	前のウィンドウへ（prev）
n または ^N または Space キー	次のウィンドウへ（next）
t	時刻とロードアベレージの表示（time）
^W	ウィンドウ一覧の表示（windows）
x または ^X	端末のロック（lockscreen）
^Z	サスペンド（suspend）
H	ログの on／off（log）
M	モニタの on／off（monitor）
Z	端末のリセット（reset）

＊6 ./etc/etcscreenrcを見ると分かりますが、tsはステイタスラインにカーソルを移動させるためのエントリ、fsはステイタスラインからカーソルを戻すためのエントリ、dsはステイタスラインの使用を停止するためのエントリです。本文中の設定は、ステイタスラインへの出入りを、タイトルバー出力のエスケープシーケンスに置き換えることでステイタスラインの代りをさせています。

【表4】Screenの主なコマンドラインオプション

オプション	機能
-c　<ファイル名>	ユーザー設定ファイル名の指定。デフォルトでは$HOME/.screenrc
-d [PID.セッション名]	指定したセッションをデタッチ
-D [PID.セッション名]	デタッチした後、親プロセスにHUPシグナルを送る
-ls [MATCH]／または-list [MATCH]	指定された文字列MATCHにマッチするセッションのリストを表示します。MATCHは省略可能
-r [PID.SESSIONNAME]	デタッチされたセッションにレジュームします。複数のセッションが存在する場合は、レジュームしたいセッション名をオプションで指定
-r SESSIONOWNER/[PID.SESSIONNAME]	他のユーザーのセッションにレジュームを試みます
-R	レジュームを試みますが、レジュームできるセッションが見つからなかった場合は通常通りセッションを作成します
-t NAME	ウィンドウのタイトルを指定します。shelltitleコマンドと同義
-v	バージョン番号を表示
-wipe [MATCH]	「死んだ」セッションを掃除します。screen -lsで(Dead)と表示されたセッションが該当します
-x	マルチディスプレイモードでアタッチ
-X [COMMAND]	指定されたCOMMAND文字列をScreenのセッションに送ります。このオプションはバージョン3.9.9に新設されました

■基本的な使い方

前置きがずいぶんと長くなってしまいましたが、いよいよScreenの利用法の解説です。この節ではScreenの基本となる操作を同時体験できる形式で紹介します。端末はktermを使います。

●起動とウィンドウ作成

では早速Screenを起動してみましょう。

```
$ screen
```

とすると、画面6のようなタイトル画面が出てきます。ここでEnterキーを押すと、シェルのプロンプトが出た状態になります(画面7)。このシェルは「Screenの中で動いているシェル」です。「ウィンドウを1つ作成して、その中でシェルを起動する」というのが、Screenのデフォルトの動作です。では、ここで何か適当なコマンド、例えばlsを実行してみます(画面8)。

その後に、^Acと入力してみましょう。すると、今まで出ていたコマンドの結果が消えて、シェルのプロンプトが出てきたと思います。^Acは画面をクリアするためのコマンドではなく、「新しいウィンドウを作成してそこでシェルを起動する」コマンドです。便宜上、最初にlsを実行したウィンドウを「ウィンドウ0」、新しく作成されたウィンドウを「ウィンドウ1」と呼ぶことにします。

では新しくできたウィンドウ1で何か別のコマンド実行してみましょう。今度は少し動きのあるコマンドということで、topにしてみましょうか(画面9)。その後に、^A^Aと入力してみます。すると、先ほどまで動いていたウィンドウ0が表示されると思います。topの動いているウィンドウ1は見えなくなってしまいましたが、topそのものは動き続けています。

では、もう1度^A^Aを入力してみます。またウィンドウ1に戻ってきたはずです。topは動き続けていますね(図1)。このように、Screenでは1つのディスプレイで複数のウィンドウを持つことができ、自由に切り替えたり増

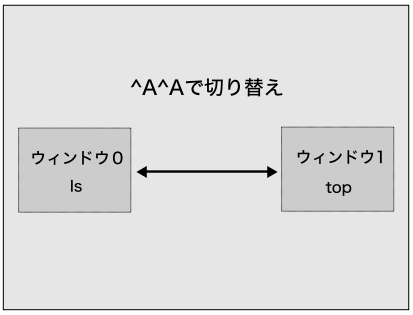
【画面6】タイトル画面

【画面7】Screen起動直後の状態

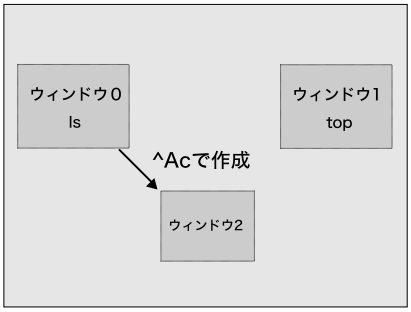
【画面8】lsを実行した状態

【画面9】別のウインドウでtopを実行した状態

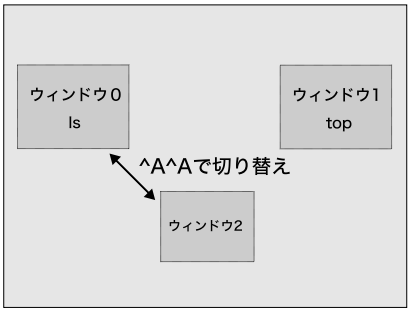
【図1】2つのウインドウが開いた状態



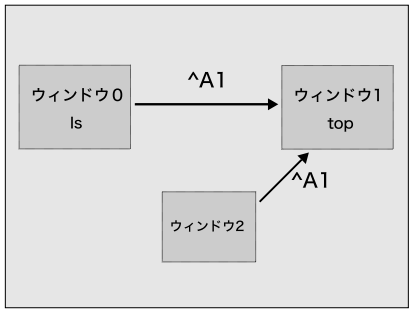
【図2A】ウインドウ2を作成



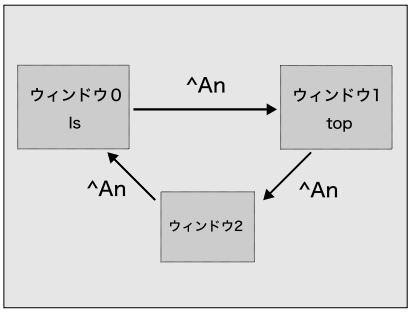
【図2B】



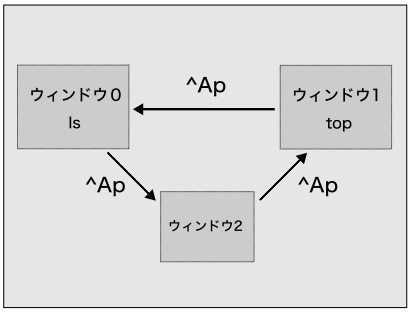
【図3】ウインドウ番号を指定してウインドウを切替



【図4】「次のウインドウ」に切り替え



【図5】「前のウインドウ」に切り替え



減させたりといったことが可能なのです。

●ウィンドウの切り替え

^A^Aを入力すると、ウィンドウ0とウィンドウ1を行ったり来たりすることが分かりました。では、ウィンドウ0が表示された状態で、さらにもう1つウィンドウを作ってみましょう。ウィンドウを作るためのコマンドは^Acですよ。できたウィンドウを「ウィンドウ2」と呼ぶことにします。この状態で^A^Aとすることで、ウィンドウ2とウィンドウ0を行ったり来たりできます（図2A、図2B）。ではウィンドウ1に移動するにはどうすればいいのでしょうか？

実はScreenでは、ウィンドウごとに番号が付いています。最初にできたウィンドウの番号は0、次から順番に1、2……となっています。例えば現在の状態では、「ウィンドウ0」のウィンドウ番号は0、「ウィンドウ1」のウィンドウ番号は1、「ウィンドウ2」のウィンドウ番号は2です。

このウィンドウの番号を利用した、ウィンドウの切り替えも可能です。^A1と入力してみてください。topの画面が出てきましたか？　もしそうなら、ウィンドウ1に戻って来ています。同様に、^A0とするとウィンドウ0、^A2とするとウィンドウ2に切り換わることを確認してください（図3）。

最初に行った^A^Aという操作は、「直前にアクティブだったウィンドウに切り替える」という操作になります。^A0でウィンドウ0に移動してから、^A2でウィンドウ2に切り替えた後に、^A^Aを押すとウィンドウ0に戻ることが分かります。

この他、「次、あるいは前のウィンドウ番号に切り替え」という切り替え方もあります。ウィンドウ0に移動してから^Anと入力すると、ウィンドウ1に切り替わります（次に切り替え。図4）。ウィンドウ2に移動してから、^Apと入力すると、ウィンドウ1に切り替わります（前に切り替え。図5）。^Apは^A^Pでもよいことになっています。また、^Anは、^A^N、または^A + Spaceキーでも構いません。

もう1つコマンドを紹介します。^Aw（または^A^W）は、ウィンドウの一覧を見るためのコマンドです。メッセージラインに、リスト3のように表示され、しばらくすると消えたと思います。この一覧の見方は、

- 先頭の数字はウィンドウの番号
- 変な記号がいくつか続いて、スペース1つあけてウィンドウのタイトル

といった感じです。今回の例では0番〜2番の3つのウィンドウがあり、タイトルはいずれも「zsh」となっています。ウィンドウのタイトルは、何も指定しなければ起動したときのシェルの名前になっています。「変な記号」については、とりあえず表5の3つを覚えておいてください。表6にここまでの操作をまとめておきます。カッコの中は対応するコマンド名です。

●セッション管理

前節で3つのウィンドウを作り、それらを1つのディスプレイの中で切り替えつつ、自由にいじることができることを知りました。今でもウィンドウ1でtopは動いているでしょうか。動いていなかったらもう1度起動してください（図6）。

さて、（少々乱暴ですが）Screenの動いているktermを閉じてみましょう。ウィンドウマネージャから「閉じる」操作をするとか、別の端末からktermを`kill`する、といった方法で無理矢理ktermを終了させてみてください（図7）*7。

無事に（？）終了したら、もう1回ktermを起動してください。もちろんここではScreen

【リスト3】

```
0-$ zsh  1$ zsh  2*$ zsh
```

【表5】

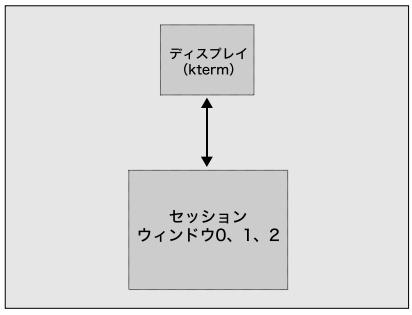
記号	意味
*	アクティブなウィンドウ
-	直前にアクティブだったウィンドウ（^A^Aで切り替えられるウィンドウ）
!	BELLキャラクタを受信したウィンドウ（そのウィンドウがアクティブになると消えます）

【表6】

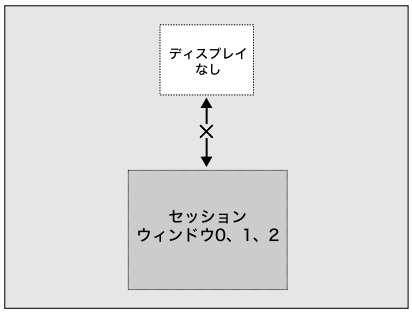
コマンドキー	機能
^Ac	新しいウィンドウを作成 (screen)
^A^A	直前にアクティブだったウィンドウに切り替え(other)
^A[番号]	指定した番号のウィンドウに切り替え(select)
^An または ^A^N または ^A + SPACE キー	次のウィンドウ番号に切り替え (next)
^Ap または ^A^P	前のウィンドウ番号に切り替え (prev)
^Aw または ^A^W	ウィンドウの一覧を表示 (window)

＊7　Screenでは、端末が終了しても中のプログラムは終了させないのがデフォルトの動作ですが（autodetach）、標準でない.screenrc（友達から貰ってきたなど）を使っている場合は、内容を確認してからこの実験をしましょう。「autodetach off」という行があった場合は、それを削除するか、「autodetach on」に変更してください。「autodetach off」の状態でktermを終了させると、中のプログラムも終了してしまいます。

【図6】ディプレイとセッション



【図7】kterm が終了した状態



【リスト4】

```
markkeys 'h=^B:l=^F:$=^E:^U=^Z:^D=^V'
```

【リスト5】

```
screen -t EMACS 0 emacs -nw
```

【表7】detach コマンド

コマンド	動作
^Ad または ^A^D	デタッチした後、Screen を終了（detach）
^AD	デタッチした後、Screen が動いていた端末に HUP シグナルを送ります（pow_detach）

は動いていません。ではここで、再度Screenを起動させます。ただし今度は-rオプション付きです。

```
$ screen -r
```

先程（強制的に）閉じられてしまったktermの中で動いていたScreenに復帰しましたか？

ウィンドウ1でtopが動き続けていたら成功です。

このように、Screenのセッションは、ディスプレイから切り離された状態（デタッチ状態）でも（止まらずに）動き続けることができ、またそのような切り離されたセッションはいつでも他のディスプレイ（もちろん同じディスプレイでもよい）に再接続（レジューム）することができます。

Screenなしでこのようなことをするのは大変困難です。nohup(1)やdisown（zshのビルトインコマンド）を使えば、端末がいなくなっても中のプログラムを動き続けますが、そのプログラムをまた別の端末につなげ直すことはできません。リダイレクトやtee(1)で出力をファイルに落とすようにしておいて、その中身を見てコマンドが終わったとか終わらないとかを判断するくらいならできるかもしれませんが、いずれも「前もって準備しとけばなんとかなる」というものでしょう。nohupやdisownを忘れていたら中のプログラムは端末と一緒に死んでしまうし、出力をファイル経由で見られるようにしておかなければ、あとはたまにpsで確認するくらいしか手がないでしょう。

しかしScreenが動いていれば、先ほどのように、いきなりktermを落としてしまうような乱暴なことをしても、すぐに別の端末からつなげ直せます。出力の結果のチェックどころか、作業の続きをすることだって可能です。infoの中で「Screenの中でたぶん最も便利な機能」といっているのも納得でしょう。

え？「『Screenの中で動かす』というのも『準備』には違いないではないか？」。そんな貴方が、「ログインしたら、あるいはktermを立ち上げたらまずscreen（-r）」となってもらうことを祈りつつ、私はこの記事を書いています。

さて、先ほどは端末を無理矢理終了させてみましたが、この他にも、ネットワーク越しにいじっていて回線が切れてしまった場合など、不慮の事故からプログラムを守ることに役立つことが分かってもらえるはずです。

【表8】

コマンド	動作
^A[または ^A^[または ^A^<ESC>	コピーモードに入ります（copy）。コピーモードではウィンドウの中で動いているアプリケーションとは関係なく、自由にカーソルを移動させられます。バックスクロールも可能です。コピーモードで Space または Enter キーを押すと、その位置がコピー開始位置になります。さらにカーソルを移動させて、Space キーまたは Enter キーを押すと、コピー終了位置が確定し、コピー開始位置から終了位置までがコピーバッファにコピーされます。
^A] または ^A^]	コピーバッファにコピーされたテキストをペーストします（paste）。コピーバッファはセッションで共通です。つまり、あるウィンドウでコピーしたテキストを別のウィンドウにペーストできます。またコピーモードではテキストの検索ができます。C-rでインクリメンタルサーチ（逆方向）も可能となっています。

＊8 ^Z を PageUp の代わりにするのは Emacs 本来の動きとは違いますが……。

また、「セッションをディスプレイから切断させる（デタッチ、detach）」コマンドもあります。ここではデタッチのためのコマンドを紹介します（表7）。いずれの場合も、デタッチされたScreenのセッションをレジュームするには

```
$ screen -r
```

とします。

●コピー＆ペースト

Screenは、各ウィンドウごとにコピー&ペーストのバッファを持っています。簡単にいうと

- バックスクロール
- コピー
- ペースト

を、Screenがサポートしているということです。ここではその触りの部分だけを紹介します。まずはコマンドの一覧（表8）をご覧ください。私はさらに、コピーモードでのカーソル移動をEmacs風にするために、.screenrcにリスト4の設定を加えています*8。コピーモードは、それだけでひとつのページャができてしまいそうなほど豊富な機能を持っています。詳しくはinfoを見てください。

●ウィンドウを閉じる

ウィンドウは、それが作られたときに起動したプログラムが終了すると、自動的に閉じられます。ウィンドウ0でexitと入力し、シェルを終了した後、^Awでウィンドウの一覧を見てみましょう。0番のウィンドウが消えていることが分かります。

```
1-$ zsh  2*$ zsh
```

また、^AK（kill）でアクティブなウィンドウを強制的に閉じることができます。

●Screenの終了

すべてのウィンドウが閉じられるとScreenは終了します。また、^A\（quit）でScreenを強制的に終了することができます。

■進んだ使い方

●起動時にウィンドウを作成

^Ac、または^A^Cで新しいウィンドウを作成できることはすでに説明しました。これらのキーにはscreenというコマンドが割り当てられています（表9）。

例えば、.screenrcにリスト5のような行を入れておくと、Screen開始時に「EMACS」という名前でウィンドウが作られ、emacs -nw

【表9】

コマンド	動作
screen [opts] [n] [cmd [args]] (^Ac または ^A^C)	新しいウィンドウを作成します。[opts]には Screenのコマンドラインオプション(の一部)を指定できます。[n]は、新しいウィンドウに付けるウィンドウの番号(0〜9)を指定できます。その番号がすでに使われていた場合は、他の番号を探して割当てられます。[cmd [args]]は、新しいウィンドウで起動したいコマンドラインとその引数を指定します。省略された場合はシェルが起動されます。

【リスト7】

```
zinnia@debian:~[17]% screen -x
There are several suitable screens on:
        302.ttyp0.debian        (Detached)
        418.ttyp4.debian        (Attached)
Type "screen [-d] -r [pid.]tty.host" to resume one of them.
```

が起動します。このウィンドウは0番に割り当てられます。もし0番がすでに使われていたら1番、1番も使われていた2番……と、空いている番号を見つけるまで繰り返します。Emacsが終了するとこのウィンドウも閉じられます。

●コマンドラインからウィンドウを作成

Screenのセッションの中で動いているシェルからScreenを起動すると、新しいウインドウを作成することができます。

```
$ screen top
```

と実行すると、新しいウィンドウを作成してtopを実行します。topが終了すると、このウィンドウも閉じられます。

●リモートからのデタッチ

screenを-dオプション付きで起動することで、すでに起動しているScreenのセッションをデタッチさせられます(画面10、図8)。

●マルチディスプレイ

Screenでは、1つのセッションを複数の端末から操作することができます(マルチディスプレイ)。さらに、複数のユーザーで1つのセッションを共有することも可能です(マルチユーザー)。ここでは、マルチディスプレイ、つまり「1人のユーザーが複数の端末から1つのセッションを操作する」方法を解説します。

まずktermを1つ立ち上げ、Screenを実行します。もう1つktermを上げて、-xオプション付きでScreenを起動します。

```
$ screen -x
```

画面11では分かりづらいですが、上のktermと下のktermは同じセッションのウィンドウを表示しています。実際に試してみると、片方のktermで何かを入力すると反対側のktermにも同じものが入力されるのが分かるでしょう。2つのktermで、同じセッションの異なるウィンドウを操作することももちろん可能です。

screen -r(レジューム)とscreen -xは、すでにアタッチされているセッションに接続できない(-r)か、接続できる(-x)か、という違いがあります(図9)。

●セッションの特定

セッションの一覧は、

```
$ screen -ls
```

とすることで、参照できます。

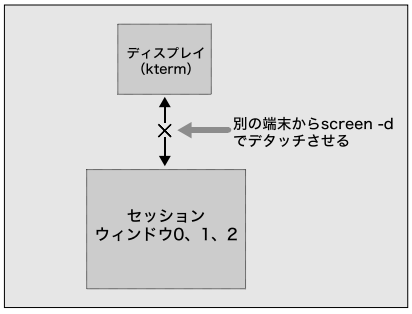
例えばリスト6では2つのセッションがあり、1つは302.ttyp0.debianという名前で、端末から切り離されています(Detached)。もう1つは418.ttyp4.debianという名前で、端末に接続している(Attached)ことが分かります。

複数のScreenのセッションがあるときに、リモートからのデタッチ(-d)、レジューム(-r)、マルチディスプレイで接続(-x)、コマンドの実行(-X)などを実行しようとすると、リスト7のようなエラーメッセージが表示されます。この場合、接続できるセッションが2つあるため、どちらかを指定しなければいけません。

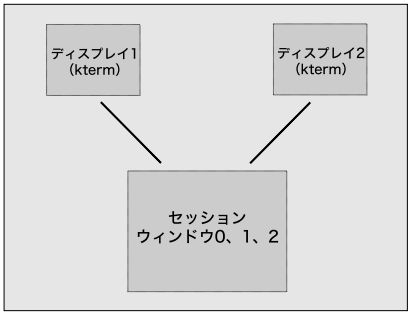
【リスト6】

```
zinnia@debian:~[26]% screen -ls
There are screens on:
        302.ttyp0.debian        (Detached)
        418.ttyp4.debian        (Attached)
2 Sockets in /tmp/screens/S-zinnia.
```

【図8】リモートからデタッチ

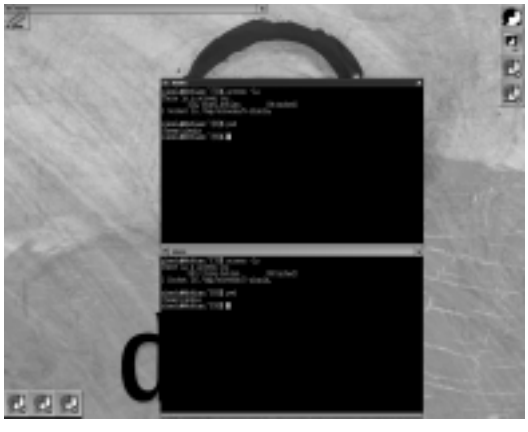


【図9】マルチディスプレイ



【画面10】左上の小さいktermから、真中のktermで動いているScreenをdetachさせた様子。[remote detached]のメッセージに注目。

【画面11】マルチディスレイの様子。上下2つのktermで同じウィンドウが表示されています。



例えば2つ目のセッション(418.ttyp4.debian)に接続したいときは、

```
$ screen -x 418
```

または、

```
$ screen -x ttyp4
```

などとすればOKです*9。

Screenのセッションの中ではSTYという環境変数が自分のセッション名を保持しています。STYが存在することでScreenのセッションの中にいることが分かり、STYの中身を見ることでScreenのセッション名が分かります。

●タイトル

ウィンドウのタイトルは以下の方法で設定できます。

・screenコマンドの-tオプションで指定

すでに説明した通り、screenコマンドは、新しいウィンドウを作成するコマンドです。-tオプションを付け得ることで、そのウィンドウのタイトルを指定できます。詳しくは前節「起動時にウィンドウを作成」をご覧ください。

・titleコマンドを使う（^AA）

titleコマンドで、ウィンドウのタイトルを設定できます。^AAと入力すると

```
Set window's title to:
```

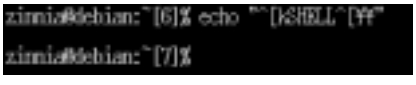
というプロンプトがメッセージラインに表示されます。ここで新しいタイトルを入力してタイトルを設定します。^A:title［設定したいタイトル］とすることもできます。

・エスケープシーケンスを使う

＜ESC＞k文字列<ESC>\という文字列を出力することでウィンドウのタイトルを変えられます（画面12）。シェルのプロンプトなどに仕掛けるとタイトルを動的に変更できます。

【画面12A】エスケープシーケンスでタイトルを変更

【画面12B】



【表10】

1	^A:passwordとして、passwordコマンドを引数なしで実行
2	「New screen password:」というプロンプトがメッセージラインに出るのでパスワードを入力
3	「Retype new password:」というプロンプトがメッセージラインに出るので、再度パスワードを入力

■Screenとセキュリティ

ここでは端末のロック、パスワードによるセッションの保護の基本的な考え方を解説します。それぞれの機能がどのような性質を持ち、それを使うことで何を守ることができるのかということを知っておくことは有益でしょう。

●ロック

lockscreenコマンド（^Axまたは^A^X）で、端末にロックをかけることができます。このときScreenは「'/local/bin/lck'」、または「'/usr/bin/lock'」を呼ぼうとします。どちらも見つからないときは、内蔵のロック機能を呼び出します。これらの呼び出しから戻るまでは端末を操作できなくなります。ただし、他の端末からscreen -d -rなどでセッションを奪うことでロックは無効になってしまいます。あくまで「端末」のロックであるということに注意してください（infoにも明記されています）。

セッションを守るためには、次節のパスワードを利用します。

●パスワード

passwordコマンドを使うことによってScreenのセッションをパスワードで保護できます。

```
password ＜パスワード＞
```

passwordコマンドに暗号化されたパスワード*10を与えることでパスワードを設定します。また、パスワードに「none」を指定するとパスワードの保護を無効にできます。さて、その「暗号化されたパスワード」ですが、作り方は表10の通りです。

パスワードが設定されると「[Password moved into copybuffer]」というメッセージがメッセージラインに出力されます。pasteコマンド（^]）を実行すると「VTlC8unoeKES6」などといった文字列が出力されるはずです。これが暗号化されたパスワードです。

あとは、.screenrcに

```
password VTlC8unoeKES6
```

と書いておけば、セッションはパスワードで守ることができます。パスワードで守られたセッションでは以下の操作は拒否されます。

・screen -Xによるコマンドの送信*11

以下の操作は正しいパスワードを入力しなければ拒否されます*12。

・screen -r
・screen -x

また、lockscreen（^Axまたは^A^X）では、ユーザーのパスワードとセッションのパスワードの両方に正しく答えないと復帰できなくなります。

■Screenのある生活

本節では、コンソールアプリケーションの紹介と、特に「Screenとの組み合わせ」にスポットを当てた使い方を紹介します。単に「こんな使い方ができるよ」というだけではありません。Screenと、その中で動くアプリケーションでできることを把握し、それを結び付けるためのアプローチを体得していただきたいということを「狙い」としています。そのため、事例の説明などが多少冗長に感じられるかもしれませんが、ご理解ください。

考え方としては、あるアプリケーションからScreenの機能を利用することで、アプリケーション単体では実現が難しいが、できたらとても便利なことを実現させるというものになります。

●Emacs、XEmacs（-nw）

ご存知の方も多いと思いますが、Emacs（またはXEmacs）を-nwオプション付きで起動すると、新しいXのウィンドウを作らずにコンソールの中でEmacsが起動します。

Xのウィンドウとして起動した場合と比べると、マウスでの操作や色使いに若干の不自由さが出てきます。しかし、どちらも全く駄目というわけではなく、設定次第である程度は使えるようになります。私のEmacs21（-nw）での画面の例は、画面13のような感じです。

Screenの中で動くEmacsは、別のウィンドウで動いているアプリケーションの外部エディタとして使うこともできます。「Emacsは便利だけど、他のアプリケーションの外部エディタとして使うには起動が重くて嫌だ」という場合は、emacsclientを使ってみましょう。

.emacsファイルに

```
(server-start)
```

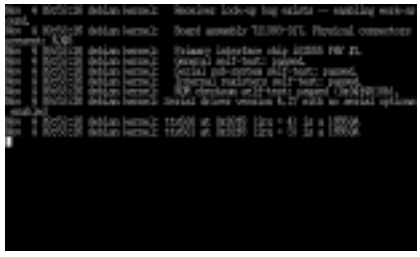
という行を追加しておきます。次に別のシェルから

＊9 -Xオプションにおいてセッションを特定する場合は-Sオプションが使えます（例：screen -S 418 -X prev）。　＊10 「暗号化（crypted）」という表現が適切かどうか……。process.c内のpass2()という関数でこの処理は行われています。　＊11 自セッション内からでも拒否するようです……。　＊12 infoのKnown Bugsにも書かれていますが、なぜかscreen -dやscreen -Dはそのまま通ります……。

【画面13】Emacs21（-nw）の様子

【画面14】emacsclientでEmacsに接続した状態

【画面15】tailで/var/log/messagesを監視している様子



【画面16】

【画面17】

```
$ emacsclient some.txt
```

としてemacsclientを実行すると、「Waiting for Emacs...」と表示されて待ち状態に入ります（画面14）。Emacs側にはsome.txtが開かれていることが分かります。編集が終わったら^X#とするとemacsclientが終了します。

emacsclientの起動は一瞬ですからアプリケーションの外部エディタとして使用してもストレスはないでしょう。私は環境変数EDITORをemacsclientにしています。XEmacsの場合はgnuserv/gnuclientという同様のサーバ及びクライアント環境が入っています。gnuclientは起動したその場でバッファを開いて編集できます。

●tail(1)

tailというコマンドを御存知でしょうか？UNIX Version 7のころからあるという「ファイルの最後の部分を表示する」プログラムです*13。

```
$ cat > file
a
b
c
d
e
f
^D
% tail -n 2 file
d
e
```

このように、ファイルの最後の数行を表示させるといった使い方ができるtailですが*14、他にも-fオプションを使うことで「ファイルの末尾を表示し続ける」ことができます。つまり、指定されたファイルの最後を表示した状態で待機して、新しい行が追加されるたびにそれを表示します。

これを使うことで、ファイルの出力を監視できるようになります。ウィンドウを、tailのために1つ上げておいて、各種ログファイル(/var/log/messages、ApacheのErrorLogやCustomLogなどなど）をtail -fで監視する、といった使い方が可能になります（画面15）。また、Screenにはmonitorというコマンドがあります。アクティブでないウィンドウの状態が変化すると下のようにメッセージラインで知らせてくれます。

```
Activity in window 0
```

このとき、ウィンドウの一覧（^A^W）を見ると、該当するウィンドウには下のように@というマークが付いています。

```
0@$ console 1*$ zsh
```

monitorコマンド、または^AMと入力することでアクティブなウィンドウのモニタモードのon／offを切り替え可能です。つまり、リスト8の設定を.screenrcに書いておくと、/var/log/messagesに新しいメッセージが来るたびに、メッセージラインとウィンドウ一覧で知らせてくれるようになります。

●w3m

w3m（[4]、画面16）は、aitoさんによるページャ及びテキストベースWebブラウザです*15。Webブラウジングをw3mメインにすることでコンソールへの依存度も増えることでしょう。とにかく1度使ってみることをお勧めします。

Screenと同様に、キーバインドと機能を細かく設定でき、工夫次第で手に馴染む自分好みのブラウザにすることができます。また、坂本浩則さんによるインライン画像表示パッチ（[5]、画面17）を使うことで画像の表示も行えます。以下、w3mとScreenの組み合わせについて取り上げてみたいと思います。

・外部ブラウザをscreen経由で呼び出す

w3mには、複数のページを切り替えつつ見るためのコマンド（SELECT：バッファ選択モード）があります。実行例2のようにしてw3mを起動した後、sを押すとバッファ選択画面が出ます（画面18）。ここでは、w3mの外部ブラウザにScreenを登録することで、w3mをタブブ

【リスト8】

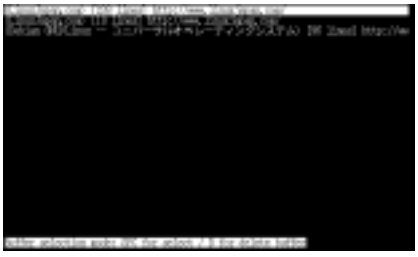
```
screen -t console tail -f /var/log/messages
monitor on
```

【実行例2】

```
$ w3m http://www.linuxjapan.com/ http://www.debian.org/
```

＊13　同様に「ファイルの先頭の部分を表示する」ためのheadというコマンドもあります。　＊14　headとtailを組み合わせると、「ファイルの任意の行から任意の行数だけ取り出す」ことができます。またheadとtailは行単位だけではなく文字単位での指定もできます。詳しくはマニュアルを見てください。　＊15　本文とは関係ないですが、私は某大学在学当時、aito先生の授業をとってました。

【画面18】w3mのバッファ選択画面



【画面19】w3mで外部ブラウザを登録している様子

ラウザ風に使う方法を解説します。

外部ブラウザは、コマンドライン引数で与えられたURLを処理できるものであれば何でも使うことができます。呼び出し方法は、w3mの（lynx風ではない）デフォルトのキーマップでは表11のようになっています。Enterキーは^Mでも入力可能ですから表12のように覚えましょう。外部ブラウザは3つまで登録できます。2番目の外部ブラウザを使いたいときは、2M、2 < ESC > Mとします。

ここで、以下のようなスクリプトを作って実行可能にしておきます。

```
#!/bin/sh
screen w3m $@
```

【リスト9】

```
#!/bin/sh
$HOME/bin/mozilla-remote -remote OpenURL\($@,new-window\)
```

【画面20】w3mで見ているページをmozilla-remote経由で表示

【表11】

キー	動作
Enterキー	リンクをたどる（GOTO_LINK）
M	外部ブラウザでリンク先を表示（EXTERN_LINK）
M-M	外部ブラウザで現在のページを表示（EXTERN）

【表13】

キー	動作
Shiftキー＋M	リンクを新しいウィンドウで開く
<ESC>M	現在のページを新しいウィンドウで開く

ファイル名はw3m-remoteとしておきましょう。これを、外部ブラウザの1番目に登録しておきます。oでオプション設定パネルを表示させて（OPTIONS）、外部ブラウザにw3m-remoteのパスを記述しておきます（画面19）。

すでに説明した通り、Screenのセッションの中でscreenを実行すると、新しいウィンドウを開いて指定されたコマンドを実行します。つまりw3m-remoteは、新しいウィンドウを開いて、指定されたURLアドレスをw3mで開くという動作をします。

このスクリプトを登録することで、表13に挙げた動作が可能になります。

なお、普段は「qでの終了時に確認する」をOnにしているけど、w3m-remoteから起動されたw3mを終了させるときにいちいち確認されるのはうっとうしいという場合は、-o confirm_qq=0つきでw3mを起動するようにw3m-remoteを修正してください。w3m-remoteと同様に、

```
#!/bin/sh
screen wget $@
```

といったスクリプトを2番目の外部ブラウザに登録しておくと、表14が可能になります。wgetが裏で動いていますからダウンロードが終わるのを待たずにw3mの操作を続けられるという訳です。wgetの代わりに後述のAriaを使うこともできます。

なお、私は3番目の外部ブラウザにはリスト9のスクリプトを登録しています（画面20）。mozilla-remoteについては「REMOTE CONTROL OF UNIX NETSCAPE」をご覧ください*16。

【表12】

キー	動作
Ctrlキー＋M	リンクをたどる（通常の動作）
Shiftキー＋M	リンク先を外部ブラウザで表示
<ESC>M	現在のページを外部ブラウザで表示

【表14】

キー	動作
2 Shiftキー＋M	リンクをダウンロード
2 <ESC>M	現在のページをダウンロード

●URLをScreenのレジスタに保存

私はSDL Watch（[7]）という、SDL関連の最新情報をお知らせするサイトを管理しています。さまざまなメールやWebページからSDL Watchで使えそうなネタを探して、一言コメントを付け加えてそれを紹介しています（ほとんどがWebページの紹介……）。SDL Watchでやっていることは、言ってしまえばそれだけのことなのですが（自虐的）、そこで面倒になるのは「URLアドレスを自分の文書にひっぱってくる」作業です。

私の場合は、

A 1番目のウィンドウではEmacsが動いており、ここでメールを読んだり文章を編集する。
B 2番目のウィンドウではw3mが動いており、ここでネタに使えそうなWebページを探す。

という状態で作業をしています。Screenのコピー&ペーストを使うと、w3m側で現在見ているページのURLアドレスを何らかの方法で表示しておいて、それをコピーしてEmacs側でペーストすることで、ある程度は楽をすることができます。例えばこんな感じです。

1. w3m側でU（GOTO）を押し、^Pでヒストリを1回たどる。これが現在見ているページのURLのはずです*17。
2. ^Aaで行頭に移動してから^A[でコピーモードに入った後、Enterキーを押して開始位置確定
3. ^Eで行末に移動してEnterキーを押すことで範囲確定→コピー
4. Emacsのウィンドウに移動して適当な場所で^A]でペースト

Uの代わりにc（PEEK：現在のURLアドレスを表示）も使えますが、この場合はコピーモードに入った後に、URLアドレスの先頭までカーソルを持っていくのが多少面倒になります。また、Uにしてもcにしても、画面に入りきる程度の長さのURLアドレスであればそれでいい

＊16　最近のMozilla 0.9.5で実装された「新しいタブで開く」（Open In New Tab）をなんとかmozilla-remoteから呼び出したいのですが、今のところうまくいっていません……。　＊17　フレームを使ったページでは必ずしもそうではないのが厳しいです。

のですが、画面に入らない長さのものでは、複数回に分けないとコピーできず不便です。

そこで、「w3m側でURLをScreenのレジスタに送りつける方法」を考えてみます。

まずは^A?でヘルプを眺めます。普段使ってる^A[、^A]は、それぞれcopy、pasteというコマンドが割り当てられていることが分かります。

マニュアルを読むと、pasteではキー（レジスタ名）を指定可能で、それが省略されると「.」というキーが使われることが分かります。copyコマンドで「.」という名前のレジスタに指定された範囲の文字列が記憶されて、pasteでその中身が貼り付けられるという流れです。さらにマニュアルの中身を「register」で検索してゆくと、レジスタに中身をセットするためのコマンドは、他にもreadreg、registerなどがあることが分かります。前者はファイルの中身をレジスタに入れ、後者は指定された文字列をレジスタに入れることができるようです。

すでに説明したように、w3mには外部ブラウザを起動する機能があります。引数にはURLアドレスを入れてくれますから、「外部ブラウザ」として、registerコマンドをうまく発行できるような仕組みがあれば目的を果たせそうです。

そこで役に立つのが、3.9.9からサポートされるようになったScreenの-X（コマンドライン）オプションです。-X付きで呼び出すと、それに続く文字列をScreenのコマンドであるとして評価させることができます。例えば、リスト10のようにすると、レジスタ「.」に「Linux Japan」という文字列が入ります。^A]と入力すると、「Linux Japan」という文字列がペーストされるはずです（画面21）。

そこで、w3m外部ブラウザとして以下のようなコマンドを登録します。

```
screen -X register .
```

リンクをMでたどると、そのリンクのURLアドレスがScreenのレジスタに入ります。また、< ESC > Mで現在見ているページのURLアドレスがScreenのレジスタに入ります。この方法を使うと、w3mで表示されているページのURLアドレスをEmacs側にペーストするには、

1. w3m側で < ESC > Mとして現在見ているページのURLアドレスをコピー
2. Emacsのウィンドウに移動して適当な場所で^A]でペースト

というふうに簡略化されてとても快適です。

もちろん、最初から「URLアドレスをScreenのレジスタに送りつける」ための機能をw3m側に実装して、専用キーに割り当てるといったことも可能です。しかし、w3m側でできること、Screen側でできることをよく知り、その中で目的の機能を実現できるかどうかを考えるというアプローチを忘れないようにしてください*18。

【リスト10】

```
screen -X register . "Linux Japan"
```

【画面21】コマンドラインからレジスタに文字列を送った様子。次の行でコマンドラインからペーストさせています。

■GUIアプリとの連動

X上の端末で作業をしている場合は、何も狭い端末の中だけですべての仕事をしようなどと考えることはありません。必要に応じて端末の外で動くアプリケーションを使いましょう。アプリケーションによっては、コマンドラインから指示を送れるものがあります。

●XMMS

XMMS（[8]）は、

```
$ xmms ファイル名
```

とすることでXMMSを起動して指定されたファイルの演奏を始めます。このとき、「設定」→「オプション（タブ）」→「複数のXMMSの起動を許可する」のチェックを外すと、すでにXMMSが起動している場合はそのXMMSに対して指示を送ることができます（画面22）。

●Aria

ダウンロードツールのAriaでは、

```
$ aria -g ＜URL＞
```

で、ダウンロードするURLを送ることができます（画面23）。w3mの「外部ブラウザ」にAriaを登録すると、

- ダウンロードの最中もw3mの操作を続けることができる
- 複数のダウンロードの状況を一目で確認できる
- 細かい設定（速度制限、PROXYなど）をGUIで行える

といった利点があります。

【画面22】XMMSで演奏するファイルをコマンドラインから指示

■その他

本稿では詳しく触れることができなかったScreenの機能を簡単に紹介します。各機能の詳細はinfoをご覧ください。

●ハードコピー

hardcopy（^Ahまたは^A^H）*19コマンドで、画面のハードコピーを撮れます。ハードコピーの結果はhardcopy.Nという名前（Nは数字）で、Screenの起動したディレクトリにセーブされます。セーブする場所を変えたいときはhardcopydirコマンドで指定してください。

●ログ

log（^AH）コマンドでログをとる、とらないの設定を切り替えます。セッション全体で

【画面23】Ariaでダウンロードさせたいファイルをコマンドラインから指示

＊18 なお、w3mでは外部ブラウザは3つまでしか登録できません。私はすでに書いた通り3つとも登録してしまっておりますので、ソースをいじって4つ目の外部ブラウザを登録できるようにしました。このくらいの変更なら自分の責任でなんとかできる範囲だと思います。　＊19　インストール時の/etc/screenrcでは、これらのバインドを無効にする設定が入っています。そのためデフォルトでは使えませんのでご注意ください。

script(1)と同様なログをとることができます。ログはscreenlog.Nという名前（Nは数字）でセーブされます。

●リージョン

Screenでは端末を複数の領域に分割し、それぞれに別々のウィンドウを表示させられます。このとき、個々の領域をリージョンと呼びます。リージョンの操作の表15に示します。

●マルチユーザモード

Screenは、複数のユーザーで1つのセッションを共有できます。これをマルチユーザーモードといいます。このモードでは、ユーザーごとに権限を設定したり、使えるコマンドを制限したりできます。マルチユーザモードを使うと、

- 発表するユーザーがセッションを作る
- 聴衆ユーザー達にアクセス権限（見るだけ）を与え、セッションに接続させる

などとすることで、発表者の操作を聴衆の目の前の端末に配信させるといったことが可能です。

【表15】リージョン操作

キー	動作
split（^AS）	リージョンを2つに分割します。新しいリージョンには何もないウィンドウ(ウィンドウ番号-）が割り当てられます。Emacsのウィンドウ操作における^X^2に相当します。
focus [direction]（^A<TAB>）	別のリージョンにフォーカスを移します。directionには、up（前）、down（次）、top（先頭）、bottom（末尾）を指定できます。省略するとdownが指定されたことになります（Emacsのウィンドウ操作における^Xoに相当）。
only（^AQ）	現在のリージョンを残して他をすべて閉じます。Emacsのウィンドウ操作における^X^1に相当します。
remove（^AX）	現在のリージョンを閉じます。リージョンが1つの場合は何も起こりません。Emacsのウィンドウ操作における^X0に相当します。
resize [lines]	現在のリージョンのサイズを変更します。
resize +N	N行追加
resize -N	N行減少
resize N	N行にする
resize =	すべての行を同じ高さになるようにする
resize max	指定できる最大の行数にする
resize min	指定できる最小の行数にする

■まとめ

いくら「Screenは便利だ」といっても、そもそもコンソールと、そこで動くアプリケーションを有効に活用してなければその「便利さ」を十分に得ることはできません。しかし、本文中でも述べましたが、コンソールだけですべての作業をやるべきだとは私は考えていません。

コンソール上のアプリケーションと、Xのアプリケーションは排他的な存在ではなく、それぞれに適した使い方があるはずです。それを理解し、コンソールでできることを増やしておいて、それをScreenの中で動かすようにする努力はとても有益だと思います。

- 端末をいくつも立ち上げなくても済むためスペースの節約になる
- Xのウィンドウの数も減るので、キーボードでの切り替えも素早くできる
- 仮想端末もキーボードで手軽に切り替えられる
- リモートからログインして操作することもできる
- 回線の切断に強い

Screenのセッションはリモートから接続しても軽快に動作します。9600bpsのモデム経由でもしっかり動きます。

■あとがき

私は普段からノートパソコンを持ち歩いています。Screenの中でメールの読み書きやプログラミングなどを行っていますが、近くにデスクトップマシンがあるときは、そこからノートパソコンにログインしてScreenのセッションをいじっています。私のノートパソコンは大きめのキーボードとトラックボールつきで、個人的にはかなり扱いやすいと思っていますが、それでもやはり使い慣れたHHK Liteやホイール付きマウスにはかないません。

ところで、私がScreenの紹介を目的としたページ（[10]）を公開したとき、周囲でScreenを使っている人はほとんど見つかりませんでした。当時の私はそのような「Screenの知名度、評価の低さ」を以下のように分析しています。

- Screenの存在を知らない人が多い
- Screenの素晴しさに気付いていない人が多い。
- Screenの便利なところを活用していない人が多い

今でもScreenのことを扱った文献やページはそれほど多くないような気がしますが、本稿がScreenの知名度向上と、皆さんのScreen生活をより豊かにするきっかけになることを祈っております。

Resource

[1] screen
http://www.gnu.org/software/screen/

[2] freshmeat II
http://freshmeat.net/

[3] JM Project によるScreen マニュアル和訳版
http://www.linux.or.jp/JM/html/GNU_screen/man1/screen.1.html

[4] w3m Home Page
http://w3m.sourceforge.net/

[5] w3m のページ
http://ei5nazha.yz.yamagata-u.ac.jp/~aito/w3m/

[6] REMOTE CONTROL OF UNIX NETSCAPE
http://home.netscape.com/newsref/std/x-remote.html

[7] SDL Watch
http://www2.jan.ne.jp/~zinnia/sdl/watch.html

[8] XMMS
http://www.xmms.org/

[9] Aria
http://aria.rednoah.com/

[10] Coming Soon
http://www2.jan.ne.jp/~zinnia/