screenの利用について

笠原, 義晃
九州大学情報基盤センター

https://doi.org/10.15017/1470575

出版情報：九州大学情報基盤センター広報 : 学内共同利用版. 2 (3), pp.180-191, 2002-11. 九州大学情報基盤センター
バージョン：
権利関係：

screenの利用について

笠原 義晃*

1　screenってなに?

近年、家庭へのネットワーク接続の普及や、ノートPCの普及により、自宅や出張先からネットワーク経由で大学の研究室や情報基盤センターのUNIXワークステーション等を利用する機会が増えているのではないかと思います。

自宅や出張先から大学に接続する場合、telnetやssh(WindowsならTeraTerm + TTSSHやputtyなど)で文字端末を開いて遠隔ログインし、作業する事も多いのではないでしょうか。そういう場合に「回線が不安定で作業中に接続が切れてしまった」「仕事の途中で接続を切らなければならなくなった」「大学の作業の続きを家でやりたい」といった問題が起こる事がままあります。バックグラウンドに落とせるような性質の仕事ならいいのでしょうが、対話的処理だとそうもいかず、接続を切ってしまうとその処理をしているプロセスも死んでしまいます。

そこでフリーソフトscreenの出番です。screenはUNIX系OSで動作し、物理的な一つの端末を複数のプロセス(通常は対話的シェル)で共有できるようにする、言うなれば文字端末用のウィンドウマネージャです。screenを使うと、一つの端末の中で複数個の仮想画面を利用でき、またその仮想画面とその中のプロセスを生かしたまま、自由に端末から切り離したり(detach:デタッチ)、再接続したり(attach:アタッチ)する事ができます。不意に接続が切れても自動的にデタッチしてくれますから、慌てず騒がず再度ネットワークに接続し、遠隔ログインしなおしてアタッチすれば、切れる直前の状態に元通り。やりかけの仕事を、違う時間・違う端末から続ける事も簡単にできます。もちろん、バックスクロールや仮想画面間のカット&ペーストもできますし、画面分割によるマルチウィンドウもサポートしています。

screen自体の歴史は古く、著作権表示には1987年の表示があります。筆者は学生の頃からscreenを愛用しており、もう10年以上使っていますが、存在を知らない人も多いようなので紹介する事にしました。UNIXでシェルを多用している人は是非一度利用してみてください。screen無しの生活は考えられなくなる事請け合いです。

2　コンパイルとインストール

この章ではscreenのインストールについて解説します。

まず、ソースのアーカイブファイルを入手しましょう。2002年10月現在、screenの

*九州大学情報基盤センター
E-mail : kasahara@nc.kyushu-u.ac.jp

最新バージョンは3.9.13です。screen-3.9.13.tar.gzを入手してください。screen本体の配布元はftp://ftp.uni-erlangen.de/pub/utilities/screen/です。かつてscreenは日本語などの2バイト文字に対応しておらずパッチが必要でしたが、現在のscreenには統合されているので特に手当する必要はありません。

アーカイブを入手したら、ソースを展開します。展開するとscreen-3.9.13というディレクトリができます。

```
% gzip -dc screen-3.9.13.tar.gz | tar xf -↵
```

展開できたら、コンパイルしてインストールします。configureが付属しているので簡単です。

```
% ./configure↵
% make↵
% su↵
# make install↵
```

これでインストールは終了です。なお、screenは個人ユーザ権限でのインストールも可能です。

3 使い方

3.1 設定ファイル

インストールは無事済みましたか? それでは実際に使ってみましょう。まず、必要最小限の設定ファイルを作ります。screenの設定ファイル.screenrcはユーザのホームディレクトリ直下に置きます。無くても使えない事はありませんが、少なくとも以下に示す漢字コードとエスケープ文字の設定はしておく事をお勧めします。

```
defencoding eucJP
escape ^Zz
```

「defencoding eucJP」は、入出力に使用する漢字コードの指定です。遠隔ログインなどで使う端末ソフト側に合わせてください。sjisやjisも使えます。日本語EUCがeucじゃなくてeucJPなのは、中国語EUC(eucCN)や韓国語EUC(eucKR)にも対応していてeucだけだと区別できないためです。

「escape ^Zz」は、screen自体へコマンドを送る時に入力するエスケープ文字の指定です。「escape xy」の形で書き、xがエスケープ文字、yがxを入力するのに使うキーとなります。xとyにはそれぞれ「文字1文字」「^に続けて文字1文字(Ctrl+文字の意)」「\に続けて8進数(その数値に対応するASCII文字として解釈)」「\に続けて特殊文字(^や\を字面通りに解釈)」が書けます。

この例では、エスケープ文字には^Z(Ctrlキーを押しながらz)を使う事にし、また^Z自体をscreenの中のプロセスに送りたい時には^Zに続けてzを入力するように設定しています。screenの標準設定ではエスケープ文字は^Aに、エスケープ文字の入力は^A aになっています。しかし、最近の高性能シェルやエディタでは^Aは「行頭に移動」の機能が割り振られている事が多く、これをscreenに奪われてしまうと不便ですので、他のあまり使っていないキーに変更する事をお勧めします。

ちなみに筆者は「escape ^\\\」で^\(Ctrlキーを押しながらバックスラッシュ)をエスケープ文字に指定しています(指定するバックスラッシュの数に注意)。Emacsで^Zを自分用コマンドキーに割り当ててしまっているのと、Emacsでは日本語入力にSKKを利用していて^\を使っていないからです。WnnやCannaなどではこのキーを変換モードの切替に使用するので気をつけてください。他に、^Tにしている例も聞いた事があります[1]。

以下の説明では、エスケープ文字を変更していない(^A)事を仮定して説明します。上の例のように違う文字に変更した人は適宜読み換えてください。

3.2　起動と終了

設定ファイルの準備ができたら、起動してみます。

```
% screen↵
```

screenのバージョンと著作権表示が画面に出力されますので、スペースキーかリターンキーを押してください。通常のシェルが起動するはずです。^Aがscreenに取られている以外は通常の操作方法と変わりません。^Aを入力したい時には^A aと入力してください。また、毎回著作権表示を見たくないという人は、.screenrcに

```
startup_message off
```

と書き足す事により抑制できます。

screenの起動によって生成されたシェルを終了すると、screen自体も終了します。

[1] transpose-chars(カーソル前後の2文字を交換)を多用する人は使えませんが。

```
% exit

[screen is terminating]
%
```

3.3 デタッチとアタッチ

通常 screen の説明ではこの後複数の仮想画面の作り方に行く所ですが、今回は screen の目玉である「デタッチ」「アタッチ」機能の説明を先にしようと思います。

デタッチ

screen を起動して、試しにそこで vi や Emacs みたいな対話的なコマンドを起動してみてください。普通に起動しますね? そうしたら、そのまま^A ^D または^A d と入力します (どちらでも同じ意味です)。

```
~
/tmp/vi.bPXe60: new file, iso-2022-jp: line 1
[detached]
%
```

[detached] という表示と共に、screen を起動したシェルに処理が戻ってきたはずです。screen のプロセスは終了したわけではなく、バックグラウンドで動いています。これがデタッチ操作です。おおまかに図で書くと図 1(次頁) のような感じになります。

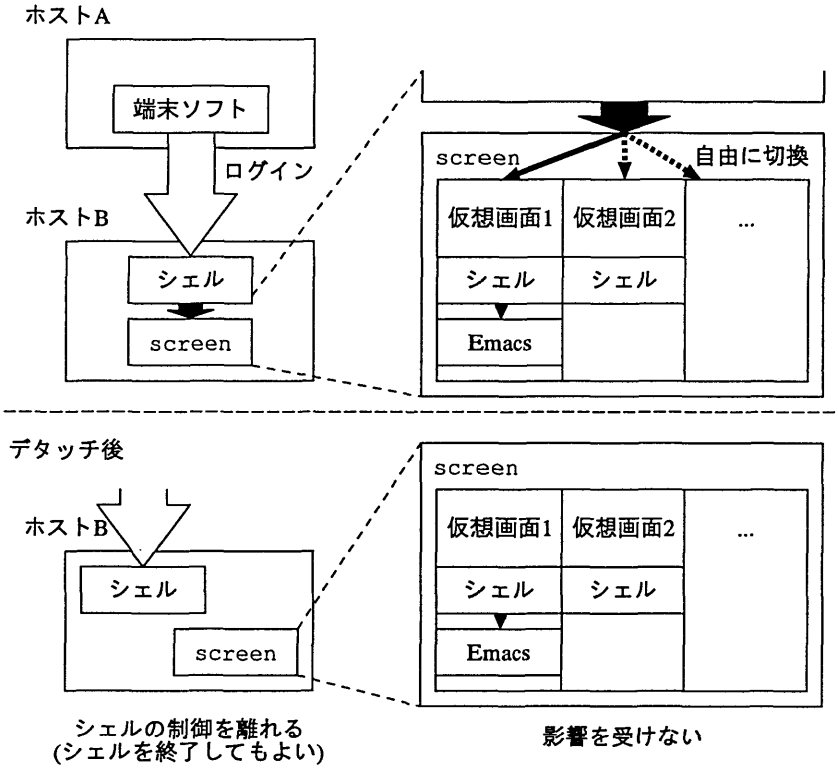
アタッチ

では、元の状態に戻してみましょう。

```
% screen -r↵
```

どうですか? デタッチする前の画面に戻ったと思います。エディタでやりかけの仕事も、デタッチした時の画面のままで復帰したはずです。これがアタッチ操作です。

図 1: screen とデタッチ



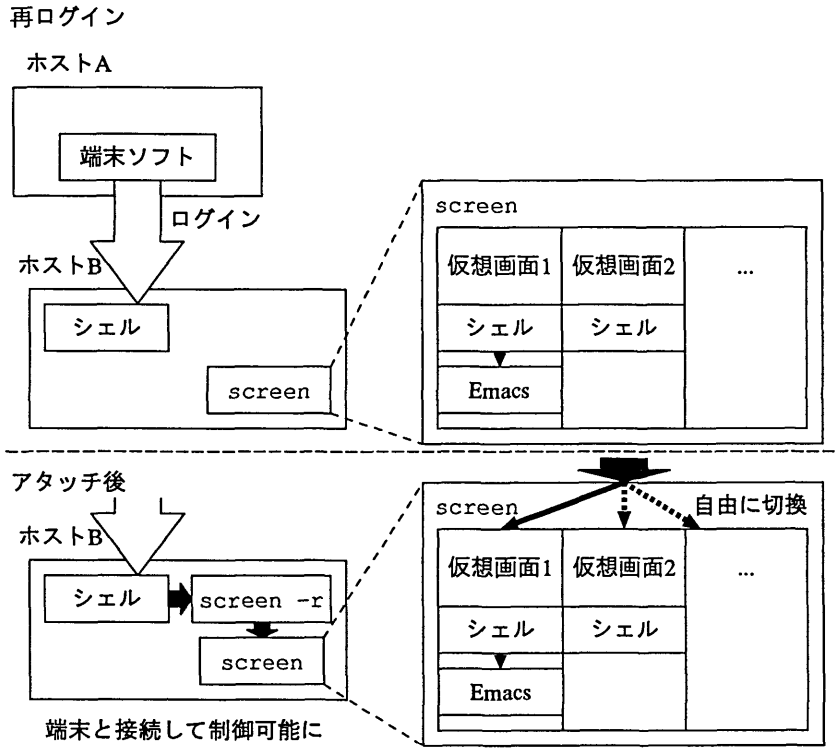
簡単でしょう? screen を使って作業中にネットワーク接続が切れてしまったりしても自動的にデタッチされますから、再度ログインして screen -r すればいいわけです。これで、ふいの接続断も怖くありません。これだけでも screen を使う価値があるというものです。もちろん、一旦デタッチしてログアウトした後、別の端末からログインしてアタッチし、作業を続ける事もできます。

おおまかに図で書くと図 2(次頁) のような感じになります。

screen 一覧

今動いている screen プロセスがいくつあるかを見るには、screen -ls と入力してください。

図 2: screen とアタッチ




```
% screen -ls↵
Your inventory:
        59164.ttyp0.elvenbow    (Detached)
1 Sockets in /tmp/screens/S-kasahara.

%
```

screen が既に端末にアタッチされていると、(Detached) の部分が (Attached) になります。

screen は同時に複数個実行する事ができて、それぞれ別々にアタッチ・デタッチできます。複数の screen がデタッチされている場合、アタッチする時にどれにアタッチするか指定する必要があります。screen -r の引数にプロセス番号を追加してください (上の例では 59164)。

リモートデタッチ

大学でscreenを使っていて、家に帰って続きを思ったけどデタッチしてくるのを忘れていた! なんて時はどうしましょう。普通にscreen -rしようとしても、既に別の端末がアタッチしているので失敗します。

```
% screen -r↵
Your inventory:
        59164.ttyp0.elvenbow    (Attached)
There is no screen to be resumed.
%
```

こういう場合は、screen -dで遠隔にデタッチする事ができます。

```
% screen -ls↵
Your inventory:
        59164.ttyp0.elvenbow    (Attached)
1 Sockets in /tmp/screens/S-kasahara.

% screen -d↵
[59164.ttyp0.elvenbow detached.]

% screen -ls↵
Your inventory:
        59164.ttyp0.elvenbow    (Detached)
1 Sockets in /tmp/screens/S-kasahara.

%
```

これで問題なく再アタッチできるようになります。接続が突然切れた時など、切れ方によってはすぐにデタッチされない事があり、そういう場合にも有効ですので覚えておきましょう。なお、screen -dを使う時にも、screenが複数動いているならプロセス番号の指定が必要になります。

マルチディスプレイモード

screenの面白い機能として、同じscreenプロセスを複数の端末で共有する機能

があります。すでに別の端末がアタッチしているscreenプロセスに対して、

```
% screen -x プロセス番号 (プロセス番号は省略可)
```

と入力する事により、既にアタッチしている端末をデタッチする事なく、追加で別の端末をアタッチする事ができます。X Windowが使える環境の人は、試しに同じホストでktermなどを2つ開き、片方でscreenを実行してもう片方でscreen -xを実行してみてください。片方の画面を操作すると、もう片方の画面も同じように更新されるのがわかるでしょう。デタッチし忘れたまま端末を離れた時に、前項で書いたリモートデタッチの代わりにこの機能を使うという手もあります。

3.4 仮想画面

デタッチ・アタッチだけでもscreenはかなり便利ですが、さらに、screenはその中に複数の仮想画面を保持し、これを自由に切り換えて使う事ができます。TeraTermなどで遠隔ログインして作業している時に、もう一つシェルが欲しくなったり、エディタとシェルを同時に使いたくなる事はありませんか? screenを利用すると、使っている端末機器に依存せず、screenのみで複数の仮想画面を使った様々な機能を利用できます。

仮想画面の作成

端末で編集作業などをしていて、急に他の事を調べたくなり、別のシェルが欲しくなる事があります。もちろん、ジョブコントロール機能や、シェルへのエスケープ機能を使う手もありますが、screenを使っているなら新しい仮想画面を作成するのが簡単です。^A ^Cまたは^A cと入力してみてください。

今まで使っていた画面が消えて、別にシェルが起動した画面に変わりました。今見ている画面が、新しい仮想画面です。この画面は今まで使っていた画面とは全く独立に動いています。仮想画面は(資源が許す限り)何個でも作成する事ができます。作成した仮想画面には順に番号がふられます(後述)。

仮想画面を作成した時に生成されたシェルを終了すると、その仮想画面は消滅し、番号は欠番になります。また全ての仮想画面が消滅するとscreen自体も終了します。

仮想画面間の移動

仮想画面を作ったのはいいけれど、前の画面に戻れないと意味がないですね。

仮想画面間の移動には以下のようなキーが用意されています。全て^A を押した後に押してください。

表 1: 仮想画面移動コマンド一覧

キー入力	動作内容
^space、^N、space、n	仮想画面を昇順に切換
^H、^P、p、backspace	仮想画面を降順に切換
数字キー (0〜9)	その番号の仮想画面に直接切換
^A(エスケープ文字)	直前に表示していた仮想画面に切換

直前に表示していた仮想画面に切り換える機能は、設定ファイルで「escape ^z^z」のように「エスケープ文字」と「エスケープ文字を入力するのに使うキー」を同じにしていると使えません。上記の例だとエスケープ文字を入力するためのキー入力が ^Z ^Z になって、重なってしまうためです。直前の端末に切り換える機能は、2 つの仮想画面を見比べる時に非常に便利なので、できれば使えるような設定にしておく事をお勧めします。

仮想画面の管理

仮想画面には作った順に番号が振られると書きました。番号の確認をするには、^A ^W または^A w を使います。

```
0*$ zsh  1$ zsh  2$ zsh  3-$ zsh
```

実際には白黒反転した表示になります。数字が各画面に振られた番号、*がついているのが現在表示されている画面、-がついているのは直前に表示していた画面です。zsh というのはその仮想画面のタイトルで、通常はシェルの名前になっています。$にも意味がありますが、ここでは割愛します。

このままでは全部同じタイトルでどれがどれだかわからないので、変更してみましょう。現在表示している仮想画面のタイトルは^A A(大文字の A) で変更できます。キー入力すると

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

というような表示が出ます。現在のタイトルの部分が編集可能ですので、書き直してリターンキーを押してください。

仮想画面の数が10を越える程になると、番号での直接移動では足らないし、^A wでも全て表示する事ができません。こういう場合には、^A "(ダブルクォート)を使うとよいでしょう。画面がウィンドウの一覧に切り換わり、カーソルキーなどで切り換え先の画面を選ぶ事ができます。

```
Num Name                                    Flags

  0 Emacs                                     $
  2 vi                                        $
  3 zsh                                       $
  4 zsh                                       $
```

3.5 バックスクロール・コピー&ペースト

ここでは、仮想画面のバックスクロールや、コピー&ペーストの方法について説明します。WindowsからTeraTermを使ったりX Windowからktermを使ったりしている場合、これらの端末ソフトにはスクロールバーがついていますし、またマウスを使ってカット&ペーストする事もできます。実際そういう機能が無い端末を使う機会というのは今ではほとんど無いような気もしますが、screenでのやり方を知っていても損はないでしょう。

コピーモードと画面操作

screenでバックスクロールやコピー&ペーストをするには、まず仮想画面をコピーモードと呼ばれる状態に切り換える必要があります。モードを切り換えてscreenの中のシェルなどに影響を与えない状態にして、カーソル移動やコピー操作を行なうわけです。

コピーモードへの移動は^A [または^A ^[(ESCキーでも可)です。画面の最下段に

```
Copy mode - Column 38 Line 4(+100) (80,24)
```

というような表示が出たと思います。数字は入力した時の仮想画面の状態によって若干異なります。Column 38 Line 4は、現在カーソルが左から38文字目の上から4行目にある事、(+100)は画面の上方にスクロールアウトした行を100行まで記憶できる事、(80,24)は仮想画面のサイズが80×24である事を示しています。

コピーモード内では、vi のカーソル移動コマンドと同様の操作でカーソルを移動する事ができます。画面の上端を越えてカーソルを動かそうとするとバックスクロールします。表 2 に主要なコマンドキーをまとめておきます。詳しくは screen のマニュアルなどを参照してください。

表 2: コピーモード内コマンド (一部)

キー入力	機能
h j k l	カーソル移動 (h から順に ←↓↑→)
0 $	行頭、行末
g G	先頭行/最終行に移動
^u ^d	半画面分ずつ上/下スクロール
^b ^f	一画面分ずつ上/下スクロール
/ ?	vi 風前方/後方文字列検索
^s ^r	Emacs 風インクリメンタルサーチ
n	直前の検索を繰り返す

コマンドキー以外のキーを押すか、ESC キーを押すとコピーモードを終了します。

```
Copy mode aborted
```

コピー

コピーモード内でカーソルを自由に動かせるようになったら、次はコピー操作です。^A [でコピーモードに入ったら、コピーしたい領域の最初の文字にカーソルを移動してスペースキーを一回押してください。

```
First mark set - Column 1 Line 5
```

最初のマークがどこに設定されたかが表示されます。これ以降カーソルを移動すると、コピー対象になる部分が反転表示されます。反転表示を見つつカーソルを移動し、コピーしたい部分が選択できたら再度スペースキーを押してください。

```
Copied 260 characters into buffer
```

コピーバッファに何文字コピーされたかが表示され、自動的にコピーモードから通常モードに復帰します。これでコピーできました。

ペースト

ペーストするには、ペーストしたい仮想画面に移動して^A] と入力してください。現在カーソルがある所にコピーバッファの内容が出力されます。ペーストされる文字列はキーから入力しているかのように入力されますので、エディタなどにペーストする時はあらかじめペーストしたい場所にエディタのカーソルを移動しておく必要があります。screenのコピーモードでペーストするわけではない事に気をつけてください。

他にも、矩形領域のコピー&ペーストや、コピーバッファに複数の部分を追加でコピーして行く操作などがありますが、ここでは割愛します。

4 おわりに

今回は、主にUNIXをシェル経由で使っているユーザを対象に、screenというフリーソフトを解説しました。screenには、今回説明した以外にも、画面分割によるマルチウィンドウ機能、画面のパスワードロック機能、バックスクロールログのファイルへの書き出し、画面のスクリーンショット記録などの機能があります。また、.screenrcでかなり細かく設定の調整が可能です。今回説明した範囲で利用していて気に入らなかったり物足りなさを感じ始めたら、マニュアルなどを参考に設定変更に挑戦してみるとよいと思います。検索エンジンで探すと日本語のページも結構見つかると思います。

screenでより快適なUNIX生活をお楽しみください。