サーバー構築/運用術

第2回 screen

先月号で説明したように，現状ではキャラクタ・ユーザー・インタフェース(CUI)を使いこなせることがシステム管理者の条件と言えそうです。管理者にとって必携のCUIツールが今月号で解説する「screen」です。時間と場所を問わずに管理作業の環境を継続することが可能になります。

（ケイ・ラボラトリー　仙石 浩明）

2001年3月3日にJavaを搭載した携帯電話の「N503i」，9日に同「SO503i」が発売されました。3月5日には「P503i」の販売が再開されたこともあり，Java搭載携帯電話のユーザー数が一気に増えそうな今日このごろです。さらに今年の夏ごろまでには，J-フォンとauからMIDP＊1を採用したJava搭載携帯電話が発売される予定であり，今年はまさにJava搭載携帯電話元年と言えるでしょう。

今後，どのくらいの勢いでJava搭載携帯電話が普及するかが焦点となりますが，普及には何よりも使いやすいアプリケーションの充実が欠かせません。そのためには開発者の人口を増やすことが最も効果的でしょう＊2。

しかし，Java搭載携帯電話用のアプリケーションを作ってみようと思っても，入門書などの情報源がまだ少なく，周囲に助けてくれる人がいないと，なかなか先へ進めない＊3のが現状ではないでしょうか。

そこで，iアプリ＊4を開発するためのオープンソース・プロジェクト＊5を開始しました。公開の場でiアプリを実際に開発することにより，開発者のコミュニティでノウハウを共有できれば，と考えています。

screen

screen＊6は，複数の端末を管理するための仮想端末マネージャです。/bin/bashなどのシェルをはじめとするキャラクタ端末上で動作するプログラムを，複数の端末から操作できるようになります(**図1**)。

例えば，職場で自分の机のWindows PCで端末エミュレータ＊7を走らせて，会社のサーバーであるUNIXホストにログインして作業しているとします。ところが仕事が佳境に入ってきたところで終電の時間が迫ってきました。ここで慌てて作業を中断してログアウトしてしまうと，明日朝に中断時の状態まで復旧させるのが面倒そうです。現在変更中のファイルがたくさんあり，どのファイルのどの部分を変更中だったかを忘れてしまう恐れすらあります。

こんなときには，screenを使うと便利です。screenが端末エミュレータとログイン・シェルの間に介在することにより，ログイン・シェルを終了させる(つまりログアウトする)ことなく端末エミュレータを切り離すことができます。従って，作

＊1　「Mobile Information Device Profile」の略。NTTドコモが独自プロファイルを採用した503iシリーズを提供しているのに対し，J-フォンとauは米Sun Microsystems社主導の業界標準規格であるMIDPを採用しました。

＊2　Windows用のソフト並に，フリーソフトウエアやシェアウエアの開発者が増えれば，きっと斬新なアプリケーションが次々と生まれるようになるに違いありません。

＊3　スタンドアロンで動く簡単なゲーム程度ならば，エミュレータ等に付属のサンプル・プログラムをいじっているうちに作れるでしょうが，せっかく携帯電話上で動くプログラムなのですから，ネットワークをフルに活用するプログラムを作ってみたいものです。

＊4　NTTドコモの503iシリーズで実行可能なJavaアプリケーション。

＊5　『ギガブルドッグ』(http://g-appli.net/developer/gigabulldog/)を参照。

＊6　http://www.gnu.org/software/screen/screen.htmlを参照。

＊7　筆者は「TeraTerm Pro」(http://hp.vector.co.jp/authors/VA002416/)を愛用しています。

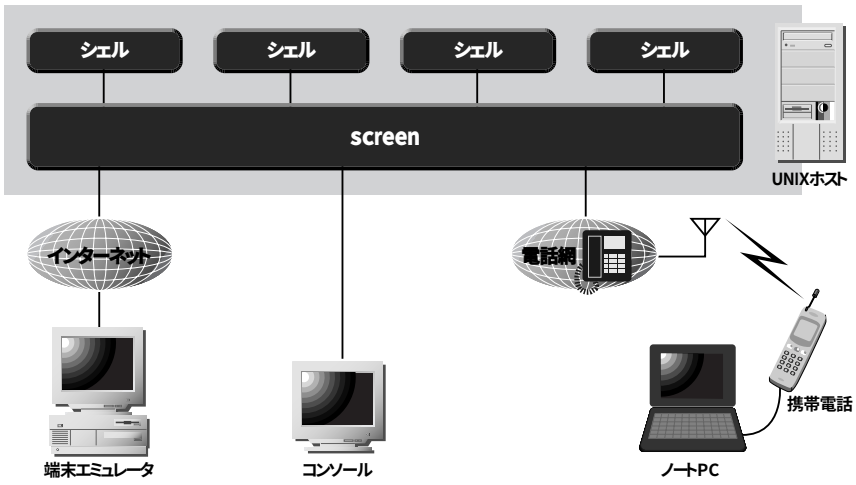

図1　仮想端末マネージャのscreen

```
asao:/home/sengoku % screen↵
```

図2　screen の起動　　**↵このマークで改行**

業を中断する際には，screenに対して端末エミュレータを切り離すscreenコマンドを与えるだけ*8です。

そして帰宅後，自宅のPCで端末エミュレータを走らせて，インターネット経由で会社のサーバーへログインし，screenに再接続すれば作業中断時点の画面がそのまま自宅のPC上に表示されるので，職場と同じ環境で作業を継続できます*9。

screenのコマンド

図2のようにコマンド・ラインからscreenを実行すると，screenのウインドウ（以下「ウインドウ」と略記）が画面一杯に表示され*10，その中でログイン・シェル*11が起動します。後はエディタなどを実行して普通に作業するだけです。

この状態は，普通のログイン・シェルと変わらないように見えますが，図1に示したようにログイン・シェルと端末との間にscreenが介在していて，C-a（コントロール・キーを押しながら「a」を押す。以下同様）に続いて英数字などのキーを押すことにより，screenに対してコマンドを送れます（これをscreenコマンドと呼ぶことにします）。主なscreenコマンドを**表1**に示します。なお，C-aをコマンド・キャラクタと呼びます。

screenにはたくさんのコマンドがありますが，よく使われるscreenコマンドにはキーが割り当てられているので，C-aに続けて1つキーを押すだけで実行できます。キーが割り当てられていないscreenコマンドの場合は，C-a :　を押した後，screenコマンド名と必要であれば引数を入力し，Enterキーを押して実行できます。

例えば，「escape XY」コマンドにはキーが割り当てられていません。C-a :　を押した後，「escape ^tt」と入力しEnterキーを押すと，コマンド・キャラクタをC-tに設定できます*12。

以下，表1に示したscreenコマンドを順に説明します。「[ ]」でくくった引数は省略可能な引数です。必要な引数を与えずに，screenコマンドを実行した場合（あるいは引数が必要なscreenコマンドをキー入力によって実行した場合），引数の入力を求めるプロンプトが画面最下行に表示されます。

デフォルトでキーが割り当てられているscreenコマンドについては，「( )」内に割り当てキーを示しました。コマン

＊8　わざわざ切り離すscreenコマンドを与えずとも，端末エミュレータ(あるいはWindows)を終了させてしまえば，自動的に切り離されます。
＊9　さらに翌朝，自宅での作業の続きを出社後に行うこともできます。あるいは，トラブル発生時には，いつでもどこでもノートPCから携帯電話経由で会社のサーバーへログインし，普段の作業環境を瞬時に再現して，トラブルの原因究明が行えます。
＊10　このウインドウには枠がないので「表示され」と言っても，画面が消去されたようにしか見えません。
＊11　Linuxのディストリビューションの多くが，bashを標準のログイン・シェルとして採用しているようです。私自身は，昔からの惰性で，cshあるいはtcshを使っています。
＊12　コマンド・キャラクタは好みに応じて設定すると良いでしょう。普段使うソフトウエアであまり使う頻度が高くないキャラクタの方が便利だと思います。筆者の場合，C-aはエディタ等で使う頻度かなりが多いため，コマンド・キャラクタをC-tに変更しています。コントロール・キー + 英字の組み合わせ以外のものも設定可能で，「^」などに設定できます。コントロール + 記号の組み合わせも可能（例えば C-^）ですが，端末エミュレータによっては，C-^などのコードを送信できない（例えばWindowsのtelnet.exe）ものがあるので，避けたほうが無難です。

ド・キャラクタを変更した場合は，最初のC-aを変更後のキャラクタで読み替えてください。

●meta （C-a a）

コマンド・キャラクタはscreenが受け取ってしまうため，screen上で実行するシェルなどのプログラムへは伝わりません。コマンド・キャラクタをscreenにではなくscreen上のプログラムに送るには「meta」コマンドを実行します。デフォルトでは，C-aに続けてaを押すことにより，プログラムへC-aを送れます。

ただし，この「meta」コマンドに割り当てられるキーは，前述した「escape XY」コマンドで変更できます。つまりこのコマンドの引数の2番目のキャラクタが設定されます。例えば「escape ^tt」を実行した場合，コマンド・キャラクタはC-tになり，「meta」コマンドに「t」が割り当てられます。この場合，screen上のプログラムにC-tを送るには，C-tに続けてtを押すことになります。

●screen （C-a c），（C-a C-c）

新しいウインドウを開きます。それぞれのウインドウには作成した順に0から数字が割り当てられます。

表1 screenコマンド

コマンド名	キー	機能
colon	:	コマンドの入力と実行
escape	XY	コマンド・キャラクタを設定
meta	a	コマンド・キャラクタを送信
screen	c	新しいウインドウを開く
select 0〜9	0〜9	0〜9 番のウインドウへ切り替える
number	N	現在のウインドウの番号をステータス行に表示する
title	A	ウインドウのタイトルを変更する
windows	w	ウインドウ一覧をステータス行に表示する
split	S	現在の領域を2つに分割する
focus	Tab	次の領域へ移る
only	Q	現在の領域以外の領域を消す
remove	X	現在の領域を消す
fit	F	セッションのサイズを画面サイズに合わせる
kanji		ウインドウ内で使用する漢字コードを指定する
detach	d	セッションを切り離す
sessionname		セッション名を変更する
lockscreen	x	画面をロックする
help	?キーに割り当てられているコマンド一覧を表示する	

●select [N]
（C-a 数字），（C-a ' タイトル）

0〜9番目のウインドウへ切り替えます。例えば，screen起動時の最初のウインドウに戻すには，C-a 0を押します。後述する「title」コマンドでウインドウにタイトルを設定している場合，C-a ' に続いてタイトルを指定できます。

●number （C-a N）

現在表示されているウインドウの番号をステータス行に表示します。例えば，X Window System上のktermなどでは，端末画面の下部外側に表示されます。ステータス行を持たない端末エミュレータの場合は，最下行に表示されます。

●title [タイトル]（C-a A）

C-a Aを押す，あるいは引数（タイトル）を省略して「title」コマンドを実行すると，**写真1**のように最下行に「Set window's title to: bash」などと表示されます。「bash」の部分が現在表示中のウインドウのタイトルです。デフォルトでは，ウインドウで起動されたプログラム名（この場合はログイン・シェル）になっているので，このタイトル文字列を適宜編集してタイトルを変更できます。

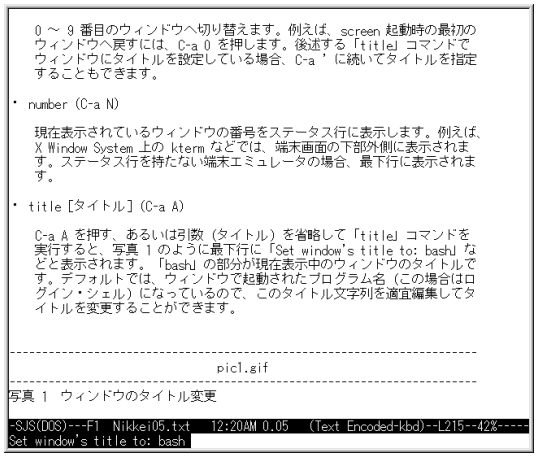

写真1　ウインドウのタイトル変更

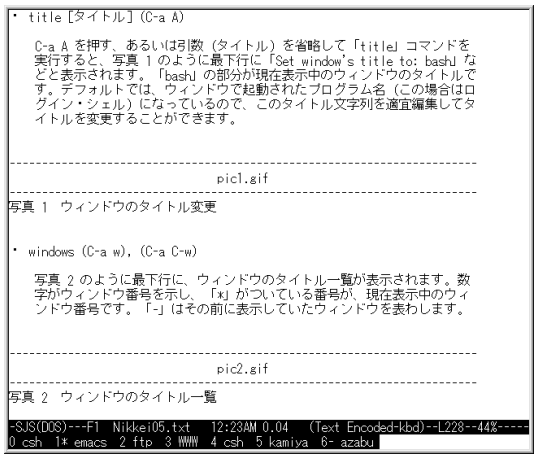

写真2　ウインドウのタイトル一覧

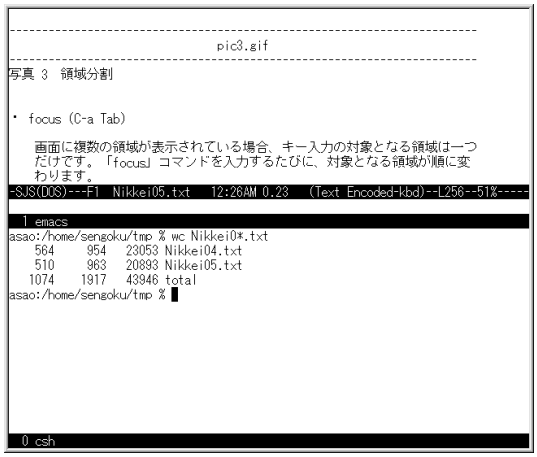

写真3　領域分割

●windows　(C-a w)，　(C-a C-w)

写真2のように最下行に，ウインドウのタイトル一覧が表示されます。数字がウインドウ番号を示し，「*」が付いている番号が，現在表示中のウインドウ番号です。「-」はその前に表示していたウインドウを表わします。

●split　(C-a S)

画面が上下に分割され，それぞれにウインドウを表示できます。ウインドウを表示できる部分を「領域」と呼びます。screen起動時のデフォルトは，画面全体が1つの「領域」ですが，「split」コマンドを入力するたびに「領域」が2分割されます(**写真3**)。

●focus　(C-a Tab)

画面に複数の領域が表示されている場合，キー入力の対象となる領域は1つだけです。「focus」コマンドを入力するたびに，対象となる領域が順に変わります。

●only　(C-a Q)

現在の領域以外の領域を消して，現在の領域が画面全体に表示されるようにします。

●remove （C-a X）

現在の領域を消します。

●fit （C-a F）

screenセッションの表示サイズを端末の画面サイズに合わせます（後述）。

●kanji ウインドウ漢字コード [端末漢字コード]

ウインドウ内で使用する漢字コードを指定します。2番目の引数を指定した場合，screenが端末に出力する漢字コードも指定します。漢字コードは，「jis」，「euc」，「sjis」のいずれかです。

●detach （C-a d），（C-a C-d）

screenと端末を切り離します（後述）。

●sessionname [セッション名]

screenセッションの名前を変更します（後述）。

●lockscreen （C-a x），（C-a C-x）

画面をロックします。解除するにはパスワードを入力します。

●help （C-a ?）

現在キーに割り当てられているscreenコマンドの一覧を表示します。

screenには，このほかにも多数のコマンドがあります。ぜひscreenのマニュアル*13を参照してください。

screenの切り離し

screenは，表1に示した「detach」コマンドを実行すると，screenから端末を切り離せます。この時の各プロセスの関係を**図3**に示します。screen上で実行していたプログラム側から見ると，「screenセッション」プロセスが，あたかも端末であるかのように振る舞います。つまり，screen上のプログラムにとっては，「screenセッション」の先に端末がつながっていてもいなくても差異はない，ということです*14。従って，screenを端末から切り離した後も，screen上のプログラムは全く影響を受けずに動き続けます。プログラムの出力によって画面が書き換えられた場合，新しい画面は「screen セッション」プロセスのメモリーに保持されます。

screenの再接続

切り離したscreenセッションを再び端末とつなぐには，**図4**に示したように「-r」オプション付きでscreenを実行します。すると，screenセッションに保持されていた画面が，端末側に表示されます。また，端末からキーを入力すれば

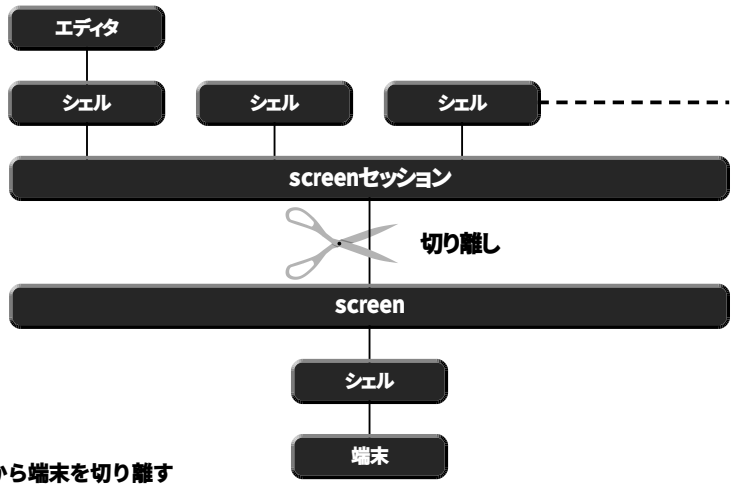

図3 screenから端末を切り離す

＊13 screenには，texinfo形式の詳細なマニュアルが付属しています。
＊14 もちろん，端末がつながっていないときはキー入力が送られてくることはあり得ません。しかし，端末がつながっていてもキー入力しなければ同じことで，screen上のプログラムにとっては，端末がつながっているかつながっていないかは判別不可能，ということになります。

```
asao:/home/sengoku % screen -r
```

図4　screenセッションの再接続

```
asao:/home/sengoku % screen -r
There are several suitable screens on:
        15819.ttyp3.asao         (Detached)
        11069.ttyp5.asao         (Detached)
Type "screen [-d] -r [pid.]tty.host" to resume one of them.
asao:/home/sengoku % screen -r ttyp5
```

図5　screenセッションが複数ある場合

```
asao:/home/sengoku % screen -r
There is a screen on:
        11069.ttyp5.asao         (Attached)
There is no screen to be resumed.
asao:/home/sengoku % screen -d -r
```

図6　接続中に，他の端末から再接続

```
[remote detached]
asao:/home/sengoku %
```

図7　他の端末からscreenセッションを奪われた端末

screen上のプログラムに送られるようになります。

1人のユーザーが複数のセッションを持つことも可能です。その場合，「screen -r」を実行すると，screenはどのセッションに再接続すべきか分からず，**図5**のように接続可能なセッションの一覧を表示します。そこで，図5のように「screen -r」に続けてセッション名を指定して実行すれば，指定したセッションへ再接続できます。セッション名のデフォルトは，tty名[*15]とホスト名の組み合せですが，これは表1に示した「sessionname」コマンドによって変更できます。

screenセッションに接続中に，ほかの端末から再接続しようとする[*16]と，**図6**のように「Attached」と表示されて再接続できません。このようなときは，「-d」オプションを追加して，screenを実行します。そうすると，再接続と同時に，接続中だった端末は**図7**のように表示してscreenセッションから切り離されます。

このままだと，だれかほかの人が勝手にこの端末を使うかも知れず不用心[*17]なので，図6において「-d」オプションの代りに「-D」オプションを使うと，screenセッションから切り離された端末をログアウトできます。

また，「-d」オプションの代りに「-x」オプションを使うと，同時に2つ以上の端末を同じscreenセッションに接続できます。それぞれ異なるウインドウを表示することもできますし，同じウインドウを表示することも可能です。後者の場合，両方の端末で同じ表示画面になります。さらに，どちらの端末でキーを入力しても，同じウインドウ上のプログラムにキー入力できます。ウインドウ上で適当なエディタ[*18]を立ち上げておけば，チャット・アプリケーションの代用になります[*19]。

同時に2つの端末を同じscreenセッションにつないだとき，双方の端末の画面サイズが異なっていると，後から接続した端末は，screenセッションのサイズと端末の画面サイズが一致しません[*20]。サイズを一致させるには，一致していない端末側で，表1に示した「fit」コマンド

＊15　screenセッションを生成したとき（つまり最初にscreenコマンドを実行したとき）のttyの名前です。
＊16　例えば，職場で端末を落とさずに帰宅してしまい，自宅で再接続しようとしたときなど。
＊17　もちろん，screenセッションを接続したままにする方がよっぽど不用心です。端末から離れるときは必ず「lockscreen」コマンドで端末をロックしましょう。
＊18　編集機能が不要であれば，catプログラムなどでも用が足ります。
＊19　もちろん，同時に発言すると混乱します。
＊20　同時に接続する場合でなければ，再接続のたびにscreenセッションのサイズが端末の画面サイズに合わせて自動的に調節されます。

を使います。

端末機能の変換

端末ごとに異なるのは，画面サイズだけではありません。画面コントロールのための，いわゆるエスケープ・シーケンスや漢字コードは，端末ごとに異なるのが普通です。UNIX互換OSでは，端末ごとの差異を吸収するためにtermcap/terminfoデータベースを持っています。すなわち，各端末の機能（カーソル移動，色設定，行挿入・削除など）ごとにその機能の有無やエスケープ・シーケンスを登録しておくためのデータ・ベースです。

UNIX上のプログラムの多くは，起動時にまず端末の種類を調べ，その端末に対応するデータをtermcap/terminfoデータベースから読み込んで，適切な画面コントロールを実現しています。ところが，普通はプログラム実行中に端末の種類が変わることまでは想定していません＊21。

そこでscreenセッションは，架空の端末「screen」＊22をシミュレートし，接続された個々の端末に合わせてエスケープ・シーケンスや，漢字コード＊23を変換します。つまり端末側から見れば，プログラムが端末に合った振る舞いをしているように見えるし，プログラム側から見れば，常に同じ端末「screen」上で動いているように見える，というわけです。

同様に，screenは漢字コード変換も行うので，例えばeuc漢字コードを表示する端末と，sjis漢字コードを表示する端末を，同じscreenセッションに接続しても，それぞれ正常に表示できます。

端末の漢字コードは，前述した「kanji」コマンドの2番目の引数で設定することもできますが，termcap/terminfoデータベースに「KJ=漢字コード」というエントリをあらかじめ登録しておけば，自動設定されます。ここで「漢字コード」は「jis」，「euc」，「sjis」のいずれかです。しかし，この「KJ=漢字コード」はscreen独自のエントリなので，システムのtermcap/terminfoデータベースに登録することに抵抗を感じる人も多いでしょう。複数の漢字コードに対応した端末も多い＊24ので，全ユーザー共通のデータ・ベースに登録することはあまり適切では無いと言えます。

そこで，システムのterminfoデータベースに登録するのと同じ効果を得るためのコマンドが，「terminfo」コマンドです＊25。

●terminfo　端末名　端末機能
[ウィンドウ側端末機能]

セッションが利用する端末機能を設定します。3番目の引数を指定した場合，screenセッション上のプログラムが利用できる端末機能も設定します。

ただし，このコマンドは表1に示したscreenコマンドと異なり，~/.screenrc設定ファイル中で指定する必要があります。~/.screenrcには表1に示したscreenコマンドも指定でき，screenセッションを生成したときに自動的に実行されます。例えば，端末「kterm」上ではeuc漢字コードで表示し，端末「vt100」上ではsjis漢字コードで表示するには，**図8**に示す行を，~/.screenrc設定ファイルに設定します。

```
terminfo kterm KJ=euc
terminfo vt100 KJ=sjis
```

図8　端末ごとに異なる漢字コードを使う場合の設定

＊21　これは当然ですね。ただし端末の画面サイズは，X Window System上のktermなど，ユーザーの操作によって変わることがあるので，実行中に画面サイズが変化することを想定しているプログラムはあります。
＊22　VT100端末互換です。
＊23　漢字コードだけでなく，欧州などの各国の文字セットの変換も行います。
＊24　例えばTeraTerm Proは「jis」，「euc」，「sjis」のいずれの漢字コードでも，表示するように設定できます。
＊25　同様に，termcapデータベースに登録するのと同じ効果を得るためのコマンドが，「termcap」コマンドです。termcapとterminfoの両方のデータベースに登録するのと同じ効果を得るためのコマンドが，「termcapinfo」コマンドです。ただし，screenはシステムにtermcapとterminfoの両方のデータベースがある場合，terminfoデータベースを参照します。ほとんどすべてのUNIX互換システムにterminfoデータベースが実装されているはずですから，terminfoコマンドを使えば十分なのかも知れません。