第 1 章

Web 付録：Linux コマンドのオペレーション教程

1.1 概要

本節では、初心者にとって「一番の山」となる Unix のコマンドライン習得について手習いをします。UNIX と LINUX は同一視して問題ありません[*1]。UNIX は商品レベルの巨大システムを構築するのにも使われるものですから、大量処理/自動処理/高速処理を可能とする様々な機能が、長い歴史の中で磨き上げられてきたツールとして提供されています。

1.2 ディレクトリ構造に慣れながら使ってみる

1.2.1 フォルダとディレクトリの階層構造

本書の教材ファイルとしてダウンロードした setupMaezono.tar.gz というファイルを人に喩えた場合、その在処を「/home/student/work/setupMaezono.tar.gz」と記するのは、「japan/ishikawa/nomi/jaist/information/maezono」[*2]といった指定の仕方に相当し、ファイルの在処を正式に規定する方策となっているので、「在処までの経路 (path) を絶対的に規定する」という事で「**絶対パス**」と呼ばれます。ある意味「ファイルの正式名

[*1] コマンドの使い方などは殆ど同じです。このコマンド体系をどうやって実装するかというプロレベルでの差異という程度の認識でよいと思います。喩えて言えば、「教習車がトヨタかホンダかは、まずは気にせずともよい」というものです。色々と乗り慣れて習熟すれば、トヨタとホンダの違いを熱く語れるようになるという感じです。

[*2] 筆者 Maezono は、Nomi(city), Ishikawa(prefecture) にある JAIST (Japan Institute of Science and Technology) に所属しており、School of Information Science の教員である。

称」と考えても良いでしょう。

「jaist/information/maezono」は、「jaist」の帳簿に載っている「school of information science」、「school of information science」の帳簿に載っている「maezono」という意味になるので、「帳簿=directory」ということで、「xxx/yyy」は、「xxx ディレクトリ下の yyy」と読み下します。maezono の居場所は、暗黙に jaist なのと同じく、「/home/student/」までは、大概いつも同じになるので、これを「～/」[*3]と略記して、「～/work/setupMaezono.tar.gz」と書くのが慣例です。

さて、そうしたらターミナルを立ち上げて

```
% cd
% pwd
```

と入力して下さい。「入力に対する応答 (返値)」として「/home/student」という表示が現れるはずです。次に

```
% ls
```

と入力すると、「Desktop/ Documednts/ ...」という返値されるはずです。

「pwd」というコマンドは、現在自分が「降臨している」ディレクトリの位置 (現ディレクトリ位置) を返してくれるコマンドで、「present working directory」の略です。Unix のコマンドは、大方このような「意味ある言葉の略号」になっていて、こういうのを**ニーモニック**と呼びます。ニーモニックを覚える癖をつければ、Unix コマンドはずっと身近になります。

「ls」というコマンドは、list のニーモニックで、「現ディレクトリ直下にあるファイルやフォルダなどを表示する」というコマンドになります。ディレクトリが「帳簿」という意味だったので、「帳簿に書かれているリストを表示する」というわけですが、どちらかと言うと「自分が降臨してる現ディレクトリから見える風景を表示する (其処にはファイルや「更に下部構造を持つディレクトリ」が見える)」というイメージで説明しています[*4]
以上、そうしたら「exit」と入力してターミナルを終了しておきましょう。

永年、コマンドラインを教えていると、「そもそもの習得が仕方がナッテナイ」という事例を幾つか経験しますので、始めに注意を述べておきます。あまりパソコンが普及していない海外などで教えていると、コマンドを一生懸命ノートに書き取って、「こないだ教

[*3] ティルダ・スラ... と読みます。
[*4] 「帳簿 A/帳簿 B」だと、帳簿 A のリストの中に更に帳簿 B があるというので、イメージがそぐわないので...

えられた通りに打ったんだけどエラーになった」といって見せられるコマンドが、スペース区切りなどがメチャメチャだったり、「今は、それが対象ファイルじゃないでしょ」というものだったりという事がよくあります。単に過去のノートの記述通りに打ち込んだりする学生が散見されますが、「作業手順や呪文として覚えるのではなく、血の通った意味をとって、自分の頭で考えてコマンドを構成しなさい」と教えています。

例えば、コマンドラインで (実行しなくて良い)

```
% sed 's/\t/\//g' temp1 > temp2
```

といった羅列が出てくるのですが、筆者も最初に此れを見たときには、「\」や「/」の羅列の何処が切れ目なのか分からなかったのを覚えています。これは (実行しなくて良い)

```
% sed 's#\t#\/#g' temp1 > temp2
```

と等価に書き換えられるのですが、「この『/』は区切りの意味 (上記で言うと#になったもの)」、「この『\』は特殊文字を表現するもの (上記の「\t」や「\/」として現れているもの)」という事を明確に訓じながらコマンドラインを実行する事が肝要です。単に「書いてあるとおりに打てばいい。これはただの作業だ」と思って取り掛かると、どうしても、「『/』の数が合ってないよ」みたいなミスに陥ります。**「何故ここにスペースがあるのか？」、「この記号の意味は何か？」をキチンと噛み砕いて**作業を進めていって下さい。

1.2.2 ディレクトリ間の移動

再度ターミナルを立て、「pwd」と入力して下さい。そうすると「/home/student」などと表示されるはずですが、これは上記の「~/」位置に相当します。つまり、通常ターミナルを立ち上げると、「~/」の位置に降臨するという事がわかります。「~/」の事を「**ホームディレクトリ**」と呼びます。次に「ls」と入力し、降臨位置からの景色を見ます。work や Documents といった項目が表示されると思います。表示される work などの項目の末尾に『/』が付されて示されているはずですが、「これはファイルじゃなくて、『その先に下部構造を持つディレクトリ』だよ」という事を意味しています。

次に

```
% cd work
% pwd
```

と入力すると (スペース区切りに注意)、今度は「/home/student/work」と返値するはずです。つまり、「cd work」によって、ホームディレクトリから見えていた下部ディレクト

リの一つである work 以下に「立ち位置を移動した」という事になります。cd は、「change directory」のニーモニックで、cd の**引数**には「移動したいディレクトリ名」を指定して利用するという事になります。

上記の操作で work 以下に移動した後、「ls」と入力すると、**現立ち位置からの景色**として、ファイルやディレクトリの一覧が見えます。「setupMaezono/」と表示される項目が見つかるはずですが、これはファイルではなく「下部構造を持つディレクトリ」だという事がわかります。そうしたら、

```
% cd setupMaezono
% ls
```

として、さらに setupMaezono ディレクトリの下に降りていってみましょう。「ls」と打って景色を見ると、us というディレクトリがあり、更にその下に california/というディレクトリがあります。更にその下に降りると、caltech/と stanford/という 2 つのディレクトリが有ります。

caltech の下に降りると、feynman というファイルが見つかります (項目末尾に『/』がない)。此処で pwd と打つと、

```
/home/student/work/setupMaezono/us/california/caltech
```

と表示されます[*5]。ls と打てば feynman が返値されますが、上記のディレクトリ故、これは「.../us/california/caltech/feynman」[*6]だという事になります。

さて、次に

```
% cd
% pwd
```

と入力してみて下さい。「home/student」と返値されますから「cd の引数を省略して入力すると、ホームディレクトリに戻ってくる」という事がわかります。本文中で「初心者はディレクトリ構造の中で迷子になりやすい」という話をしましたが、いざ迷子になっても、「**引数なしの cd**」という印籠を繰り出せば、ホームディレクトリに戻ってこられるというわけです。これで安心してディレクトリ内をふらつき回れます。

[*5] バックスラッシュ'\' の使い方については、Acronym の「長くて一行に収まりきらないコマンドの記法」を参照。

[*6] R.P. Feynman はノーベル賞物理学者で、米国カリフォルニア州にある CALTECH(California Institute of Technology) の教授をつとめた。

そうしたら、再度、~/work/setupMaezono/ まで降りて、今度は「us/california/stanford/bloch」[*7]の居るディレクトリまで降りていってみましょう。降りきったら、再度「cd」とだけ打って、ホームディレクトリに戻り、もう一度、「us/california/stanford/」まで降りてみて下さい。。。と言われると、タイピングに慣れていない初心者であれば、少し嫌気がさすのではないかと思います。毎度毎度、california だの stanford だのをミスなく打ち込むのは本当に疲れる事です。そこで、今から「**最も重要な事**」を習得してもらいます。cd と打ってホームディレクトリに戻った上で、「cd De」とだけ打って、頑張って左手の小指で tab キーを押してみて下さい。自動的に「cd Desktop」と補完されるはずです。これを「**タブ補完**」と呼びます。エンターを打ってから、再度「cd」と打ってホームディレクトリに戻ります。

上記では「cd De」まで打ってタブ補完させましたが、今度は「cd D」まででタブ補完させてみてください。タブを 1 回打っただけでは補完をしてくれない事に気づきます。タブを 2 回以上連打すると、「D から始まるディレクトリ達 (Documents, Downloads)」が表示されて、再び「cd D」が打ち込まれた状態に戻ってくると思います。タブ補完は、「打ち込んだ文字列まで」の中から候補を絞り、それが唯一になると補完を掛けてくれます。唯一になるまでは**タブを連打**して候補を見ながら絞り込んでいってください[*8]。そうしたら、再度、ホームディレクトリに戻り、タブ補完を使って work ディレクトリ以下に降ります。

work 以下に降り立ったら、「ls」と叩いてフォルダ内容を確かめ、今度は「cd s」とタブ補完して「setupMaezono」に降りて「ls」で内容を確かめ、更に「cd u」でタブ補完して us 以下に降りてみて下さい。そうしたら、cd でホームディレクトリに戻り、同じことを数回繰り返してタブ補完を指で覚えましょう。タブ補完に十分習熟した上で、再度、「us/california/caltech」の下に降りておいて下さい。

[*7] F. Bloch はノーベル賞物理学者で、スタンフォード大学で教授をつとめた。

[*8] 携帯で入力する際に同じような事をやっていると思います。例えば'Hello' と打ちたい時、'Hel' まで打てば候補が絞られて'Hello' が表示されます。

タブ補完はミスなく高速にタイピングする上で極めて重要なスキルです。ワープロ入力に慣れている方であれば、「左手小指によるタブ操作」は慣れているかもしれません。昨今の高校生になると、スマホの普及で再び、キーボードが打てない生徒が増えてしまったため、このタブ補完は、すぐには身につかず、高校生対象のサイエンスキャンプなどでも、半日くらいは、「ほらダメだよ、そこタブ使って！」としつこく言われ続けることになります。タブ補完が身につくと、タイピング作業効率が一気に高まり、高校生は順応性がさすがに高く、3 日間のサイエンスキャンプを終えると、「もうマウスなど使いたくない」というようになります。

そうしたら今度は、「ファインマンの居るキャルテック」から「ブロッホの居るスタンフォード」に移動してみて下さい。教えた事だけを使うならば、「引数なしの印籠」たる cd を繰り出してホームディレクトリに戻り、再度、work 以下に降りて行くという事をするわけですが、これでは、「キャルテックからスタンフォードに行くのに、毎度毎度、米国外から米国入国して出掛けていくようなもの」です。まあ、そうは言っても、今は此れしか知らないので、キーボードの練習と思って、まずは、そうやってスタンフォードに移動して見て下さい。

そしたら、ここから今度は、もう少しマシな方法でキャルテックからスタンフォードに移動するとしてみましょう。常識的な行き方としては、キャルテック (california/caltech) から、1 段上位の california に戻り、そこから今度は stanford(california/stanford) に降りるのが普通でしょう。この際、「1 段上位に戻る」というコマンドは、「cd ..」となります (cd の後のスペースに注意)。cd の引数に「..」(ドット 2 つ連続) を用いると「1 つ上位に」という意味になるのです。「.」(ドット 1 つ) は「現ディレクトリ」を意味し、「..」は「1 段上位のディレクトリ」という意味として使われます。それでは「cd ..」と打って、pwd で立ち位置を確かめてみて下さい。これを使って feynman と bloch の街を行ったり来たりしてみて下さい。十分飽きたら「cd」とだけ打ってホームディレクトリに戻って下さい。

次に、ホームディレクトリから、

```
% cd work/setupMaezono/us/california/caltech
```

と打ち (タブ補完！)、「ls」で現地点を確認すると「CALTECH の Feynman」の所に行き着いたことがわかります。此のように、「ディレクトリを一段一段降りていかなくても、目的地が分かっているのであれば、一気に目的地を指定して飛び移ることが出来ます。この方式を使えば、caltech から隣の stanford に行くのに、「『cd ..』としてから『cd stanford』」としなくても、「cd ../stanford」(一つ上がった所から見える stanford) とし

て一発で跳んでいけます。あるいは、現位置からの相対位置関係を使わずに、

```
% cd ~/work/setupMaezono/us/california/stanford
```

と明示的に位置を指定して跳ぶことも可能です。タクシーに乗るのに、「其処の角まがった stanford」といって頼むか、住所をカーナビに打ち込んでもらって頼むかといった違いです。

「caltech ディレクトリに移動せよ」などと指示されたときには、必ず、

```
% pwd
   /home/...[Omitted].../us/california/caltech
% ls
   feynman
```

と pwd や ls を使って、「自分が意図したディレクトリに移動しているかどうか」、「自分が仕事をする前の作業ディレクトリの状態」を確認する癖をつけるようにして下さい。

1.2.3 ファイルの操作

stanford のディレクトリに移動して「bloch」というファイルの中味を見てみましょう。「**cat** bloch」とすると、ファイルの内容が表示されます[*9]：

```
% pwd
   /home/student/work/setupMaezono/us/california/stanford
% ls
   bloch
% cat bloch
   ...(冒頭省略)
   192.168.0.250   i11server250
   192.168.0.251   i11server251
   ...(以下省略)
```

[*9] バックスラッシュ'\' の使い方については、Acronym の「長くて一行に収まりきらないコマンドの記法」を参照。

上記の表示のように pwd から書いておくと、「この場所に、このファイルがあるけど (ls の表示内容)、その中味を見ると...」という情報が相手に的確に伝わります。サーバの管理や利用時には、「エラーに見舞われた」といって習熟者に相談したり、他者に色々と操作を指示したりする場面が多々ありますが、言葉を重ねなくても、**適切なコマンド入出力を連ねて伝えれば意図が一発**で伝わります。初心者を指導していて生じるミスの殆どは、「言われたとおりにコマンドを入力したがエラーがでました」といわれて、エラーメッセージを見ると「... not found」となっているという顛末のものです。想定されているディレクトリと違う場所で作業しているために、「そのような対象ファイルが存在しない」というエラーが生じているのです。

さて、件の bloch というファイルの冒頭には何が書いてあったでしょうか？マウスを使ってスクロールして見てみると、ファイルの内容が長大だという事がわかります。cat で表示すると一気に表示が進んでしまい、よっぽどの動体視力がないと冒頭を見逃してしまいます。そこで、cat に替えて

```
% more bloch
```

としてファイル閲覧をしてみましょう。そうすると、冒頭から表示が始まり、リターンキーで 1 行毎、スペースキーで数行毎に表示を徐々に進める事が出来、ファイルの末尾に至ると表示が終了します。

ところで、この bloch というファイルが、あまり人に見られたくない内容の、しかも長大なファイルだったとして、突如、背後に人が現れたとしたら、どうしたら良いでしょうか？ファイル表示をやめたいのに、スペースキーを押しても押しても長大すぎて、人に見せられない恥ずかしい内容が延々と続き半泣き状態になってしまいます。この場合には、「**Ctrl+C**」を使えば**強制終了**することが出来ます。

尚、ウィンドウズ機の影響なのか、これを「**Ctrl+Z**」と**間違えて行う初心者**が多いので、ここで両者の違いについてキチンと述べておきます。もう一度、「more bloch」としてファイルを立ち上げ「Ctrl+Z」としてみて下さい。先程の「Ctrl+C」と同様に表示は消えますが、「[1]+ Stopped...」と表示されるはずです。そうしたら、「fg」(foreground のニーモニック) と打ってみると、再度、表示が復活するはずです。「Ctrl+Z」は強制終了ではなく**ペンディング (一時保留)** で、fg とするとペンディングされたプロセスが復帰します。ペンディングでも一応、画面から消えるので、初心者が此れを強制終了と勘違いして頻発させてしまい、**一時保留されたプロセスが大量に溜まってしまってシステムに負荷**を掛けるというトラブルがよく見受けられるので注意が必要です。

再度、stanford ディレクトリに移動して、

```
% cp bloch knuth
```

を実行した後、ls と叩いて[*10]、新たに knuth というファイルが出来ている事を確認して下さい[*11]。「**cp (ファイル名 1) (ファイル名 2)**」というのは、「(既存のファイル名 1) を、(ファイル名 2) という名前のファイルに複写」という意味で (cp は copy のニーモニック)、「(コマンド) (第 1 引数) (第 2 引数)」という形式をとります。上記によって、knuth のファイル内容は、bloch の内容と同じになっているはずなので、more で確認してみて下さい。

上記の確認が終わったら、今度は同じ stanford ディレクトリ下で、

```
% mv knuth knuce
```

としてから ls コマンドを叩いてみて下さい。先程の knuth というファイルが knuce というファイル名に変わります (ファイルの中味が変わっていない事を more で確認してみてください)。mv というコマンド (move) は 2 つの引数を取り、「第 1 引数のファイル名を第 2 引数のファイル名に変更」という意味になります[*12]。mv = move という語感から、ついつい cd (=change directory) と混同して、cd とタイプすべき場所で mv としてしまう初心者が多いので注意です。次に、同じディレクトリで

```
% rm knuce
```

として、knuce ファイルが削除されたことを確認します。rm は remove/削除のニーモニックです。

cp コマンドなど諸々は、引数にとるファイル名を現作業ディレクトリ直下のファイル名に限定することなく、

```
% pwd
    /home/student/work/setupMaezono/us/california/stanford
% cp bloch ../caltech/pauling
% ls ../caltech/
% more ../caltech/pauling
```

などと離れたディレクトリから指定する事も出来ます (「タブ補完」してますか？)。上記の more 指令は、

[*10] コマンドを入力することを「コマンドを叩く」という言い方をします。とはいえ昨今、公共の場所での問題ともなってきますので、静かなタイピングに心掛けましょう。

[*11] D.E. Knuth はスタンフォード大学に所属する数学者で、TeX の開発者としても著名。

[*12] もう、こういう書き方をしても大丈夫ですね？

```
% more ~/work/setupMaezono/us/california/caltech/pauling
```

と等価です。但し、「../caltech/pauling」の指示の仕方は[*13]、「現作業ディレクトリが同じ california 直下にある場合」しか通用しませんが、後者のディレクトリ指示の場合には、自分がどのディレクトリに居ても通用します。この文脈では、前者を「**相対パス指定**」、後者を「**絶対パス指定**」と呼びます。

次に、california ディレクトリ直下に移動し、そこで stanford や caltech のディレクトリが見えることを確認の上、

```
% mkdir berkeley
```

として ls コマンドを叩き、新たに berkeley ディレクトリが作成された事を確認しましょう[*14]。mkdir というコマンドは「make directory」のニーモニックで、第 1 引数で指定された名称のディレクトリを新たに作成するというコマンドになります。そうしたら同じディレクトリ下で、

```
% cp stanford/bloch berkeley/
```

と入力し (スペースに注意。タブ補完利用すれば「/」は打ち込まなくて自動入力されます)、「ls berkeley/」として、バークレーディレクトリ下に bloch というファイルが複写されている事を確認します (適宜、more などで内容を確認)。

次に、今、新規で作った berkley ディレクトリを削除してみましょう。削除コマンドは rm でしたが、「rm berkeley」としても削除出来ません。berkley はファイルではなくディレクトリなので、**オプション「-r」**を附して、

```
% rm -r berkeley
```

とするとディレクトリごと削除出来ます[*15]。次に、自身が california ディレクトリ直下にいる事を確認の上、「cd ..」で一段上のディレクトリに移動し、

```
% cp -r california newyork
```

として、california ディレクトリを丸ごと、newyork ディレクトリにコピーします。ここでも「対象をディレクトリとする」という意味のオプション「-r」が使われています。

[*13] L.C. Pauling はノーベル賞受賞者で、CALTECH に在職した。

[*14] カリフォルニアに、カリフォルニア州立大学バークレイ校が存在する。

[*15] 「ディレクトリごとの」というアクションに何故「-r」というニーモニックが使われるかについて調べて理解してみましょう。

次に、~/work/setupMaezono/us ディレクトリ直下にいて、california や newyork のディレクトリが見える事を確認の上、次を注意深く正確に入力し実行してください：

```
% cp california/caltech/feynman .
```

最後のドット「.」と、その前のスペースに注意してください。第 1 引数が「california/caltech/feynman」、第 2 引数が「.」となりますが、「.」というのは現ディレクトリという意味だったので、上記のコマンドは、「california/caltech の feynman を『此処に』コピーする」と訓ぜられるものになります。実際に ls コマンドで、現ディレクトリに feynman のファイルが出来ている事を確認したら、rm で此のファイルを消去し、ついでに newyork のディレクトリも消去しておいて下さい。

1.2.4 ファイルの編集

cd と打ってホームディレクトに戻り、

```
% emacs -nw work/setupMaezono/us/california/caltech/feynman
```

と入力して下さい。emacs というのはファイルを編集するためのエディタで、上記は、このエディタを使って feynman というファイルを開いて編集するというコマンド指示です。「-nw」というオプションは「no window」という意味で、新たにウィンドウを開くのではなく、ターミナル上で開くという意味です。

ファイルが開いたら、まず最初に「どうやって閉じるか」を確認しておきます。「Ctrl+X」と打ってから、「Ctrl+C」と打つとエディタが終了します。尚、これは、Ctrl を押しながら「xc」と続け押ししても同じですが、あとで、「Ctrl+X と押して**一旦離してから** i だけを打つ」といった操作が出てきて話が厄介になるので、初心者のうちは「『Ctrl+x』してから『Ctrl+c』と思って操作するのが無難です[*16]。

[*16] こうした操作は、習熟すると頭ではなく「指が覚え」ます。そのため、習熟者でも初心者に教える時に、「あれ、どうだったっけ？」と自分の指に訊くことがよくあります。珠算に長けた人が指を動かして計算するのと似ていますね。

本書では emacs エディタを利用しますが、もう一つ代表的なエディタに **vi** というものがあります。PC 利用者が Mac/Windows 派で大分されるように、エディタについても emacs/vi と好みが分かれ、こういう事を熱く論じるのが好きな人たちも居ます。本格的なサーバ運用を目指すなら、いつかは vi エディタに習熟しておく事がオススメです。vi はシステムに負担を掛けないコンパクトなものなので、どのマシンにも必ず入っているのに比べ、emacs はネットワークインストールでわざわざ導入したという事からも分かるように、比較的重いアプリとなるので、「どんな環境でも使える事」を重視するプロ視点で敬遠される一つの理由になっています。ただ、vi は古い時代のエディタの使い勝手、つまり、矢印キーやカーソル位置での入力といった直感的な操作性以前の、「X 行目の Y 文字目を消去せよ」というコマンド体制を引きずっているので、文字消去一つとっても、初学者にとって辛い事になりかねないので、ここでは直感性に長けた emacs を利用しています。可能であれば vi エディタについても、基本的な利用だけは、いつかは習得されるとよいでしょう。

そうしたら、再度「emacs -nw ...」で「feynman」ファイルを開きたいのですが、ここで、上矢印キーを 1 回だけ叩いてみて下さい。そうすると**直前のコマンドに繰り上がる**事がわかります。同じように何度か上矢印キーを叩いていくと、これまでに行ったコマンドが次々と繰り上がることがわかります。行き過ぎたら、下矢印キーを叩いて繰り下がる事が出来ます。では、この繰り上がりを利用して、もう一度、feynman ファイルを開き、今度は開いた所で、矢印キーだけ使ってカーソルを移動させてみて、「Ctrl+X」→「Ctrl+C」でファイルを閉じてみて下さい。

次に

```
% history
```

と入力してみて下さい。返値として、これまでに繰り出したコマンドがずらーっと附番されて並びますが、ここで先程の

```
% emacs -nw work/(...omitted...)/caltech/feynman
```

に附番された番号を特定し、その番号が例えば 63 だったとするならば、「!63」と入力してみて下さい。そうすると件の「feynman」ファイルが再度立ち上がると思いますが、このように、history コマンドで、自分が「これまで何をしてきたか」を確認して、繰り上がって過去のコマンドを番号で特定し再実行することも可能です。

さて、此処までは「開いて閉じる」だけでしたが、いよいよファイルの編集を行ってみましょう。まず emacs でファイルを開くと、左下の白帯の中にある「-UUU:----F1」と

いう文字列が確認できると思います。次に、ファイルの冒頭にカーソルをおいてリターンキーを押し、ファイル冒頭に空行を作ってみます。そうしたら、その行に自分の名前 (例えば student) を書き込んでみて下さい。此処で先程の白帯の文字列を見てみると、「-UUU:**--F1」と、*(アスタリスク) が表示された状態に変化した事がわかると思います。このアスタリスクは、「ファイルに変更が掛かっていて、まだ未保存」という状態を表しています。

変更内容を保存するには、**「Ctrl+X」→「Ctrl+S」**という「保存操作」を行います。そうすると先程の白帯に「Wrote...」と表示が出て、件のアスタリスク「**」が消えた事が確認できます。これで変更が保存された事になります。一度ファイルを閉じて再度開いてみて、自分の名前がキチンと文書に保存されている事を確認してみて下さい。

そうしたら次に、書き込んだ自分の名前を消して、ファイルを最初の状態に戻したあと、再度「Ctrl+X」→「Ctrl+S」で保存してみましょう。次に、もう一度、同じように冒頭行に自分の名前を書いてみて下さい。1 行目の最後にカーソルがある状態から、**「Ctrl+A」**としてみて下さい。カーソルは行頭に移動します。そうしたら次に**「Ctrl+E」**とすると、カーソルは行末に移動します。何度か繰り返してみて指が慣れたら、もう一度「Ctrl+A」で行頭に移動し、今度は**「Ctrl+D」**を使ってみて下さい。デリートキーで「後ろまで移動してから消す」だけではなく、「Ctrl+D」で前から消すという事を覚えると作業は効率化します。消してしまった自分の名前を、再度書いてみて、「Ctrl+A」で行頭に戻り、今度は**「Ctrl+K」**としてみて下さい。これは「行内のカーソル以降を一気に消す」というショートカットです。もう一度「Ctrl+K」とすると空行自体が消えます。

次に開いているファイルに、友人の名前を含めて 2 行以上の内容を記載してみましょう。追記の都度、「Ctrl+X」→「Ctrl+S」で保存する癖をつけましょう。次に、カーソルを文頭に移動させます。そうしたら Esc(エスケープキー) を 1 回押して指を離した後に、「shift+>」と押してみて下さい[*17]。カーソルは文末に跳びます。「1 回押してから『指を離す』」という事や、「> はシフトを『押しながら』キーを押す」という操作になりますので、初心者にとっては「指が覚えないうち」は、思い通りの操作が出来ない場合も多いです。もし、変な操作ミスをしてしまい、どうにも編集に復帰できなくなった場合には、**「Ctrl+G」**を何度か繰り出せば元に戻る事が出来ます (これも覚えておくべき『お守り』です)。指がキチンと上記の「**Esc+**>」を覚えたら、今度はカーソルが文末にある状態で、「Esc+<」と押してみます。そうするとカーソルは文頭に跳びます。これで文頭/文末/行頭/行末への移動を指が覚えるまで少し練習を繰り返してみましょう。

最後に Undo のやり方ですが、「(Ctrl)+(shift)+(-)」となります。適宜、編集中の文書

[*17] 『> を打つ』という操作に『シフトを押す』が含まれているので、「shift+」は表記から省略します。

で試してみて下さい。以上は emacs エディタでの最低限の**ショートカット**です。「emacs ショートカット」で検索を掛けると、もっと色々と詳細情報を知ることが出来ますが、ショートカットは最初面倒でも是非、**ちょっとした時間をケチらずに覚えてみる努力**が肝心です。「その覚える数秒/数分の手間」が、**後の何年分にも相当する効率化**になります[*18]。

1.3　より便利に使う

1.3.1　エイリアスとエイリアスファイル

Linux 初心者にとっては、「コマンドを覚えるのが面倒」(タブ補完は便利だが) という印象が芽生えてきているのではないかと思いますが、習熟者とて膨大なコマンドを全て覚えて使っているわけではないのです。実は、よく使うコマンドの組み合わせを「登録」して、自分仕様で便利に利用する事が出来るのです。

例えば、先程出てきた「emacs -nw」には「-nw」のオプションが付いていますが、毎度、これを覚えていて、いちいちタイプするのはタイピング量も増え、かつ、覚えなければならないオプションも増えて負担です。例えば、

```
% alias enw="emacs -nw"
```

のように入力した後、「enw」と叩いてみて、実際に「emacs -nw」として機能する事を確認して下さい。このような「コマンドの再定義」を「エイリアスをかける」という言い方をします。他に「『cd ..』で一つ上がって『ls』にて内容を確認する」といった操作も頻発しますが、これも、

```
% alias cup="cd .. ; ls"
```

とすると、「cup」と一発叩くだけで実現する事ができます。「(コマンド 1);(コマンド 2)」とコロン (;) で繋げると、2 つのコマンドの連なりを 1 つのコマンドとして登録する事が出来ます[*19]。

さて、ここで一旦「exit」コマンドを入力してターミナルを終了させます (「ターミナ

[*18] 職場で人を教えるのも同じですね。人に教えるのは手間が掛かるので「人に教えるより自分がやった方が速い」と、ついつい教育をサボりがちですが、最初に余分な時間を掛けて人に教えておくと、数日後には 2 馬力になり、長い目で見れば大きな作業効率化になるというものです。

[*19] ここで例示した'enw' や'cup' は実際には使いません。§2.1.4 で設定したエイリアスには夫々、'mnw', 'c' として設定されており、この実習時点でも使うことが出来ます。ここではエイリアスの仕組みを説明するため、エイリアスに設定されていない名称'enw' や'cup' を使って説明しています。

ルを exit で抜ける」という言い方をします)。そうしたら、再度ターミナルを立ち上げてみてください。ここで、先程の「enw」や「cup」が効くかどうか試してみて下さい。「そのようなコマンドは見つからない」というエラーが吐かれるはずです。折角エイリアスを設定しても、ターミナルを終了するたびに、その設定は忘却され、毎度毎度エイリアスを設定しなおす必要があるのです。では、ログインするたびに大量のエイリアスを、その都度打ち込んで設定するのかというと、そんな手間は、とても非現実的です。実際の運用では、「登録したい一連のエイリアス群」を、どこかのファイルに書いて置いておき、ログインの度に、その設定をシステムに「呑み込ませ」て反映させるという方策を採ります。

ここで「~/work/setupMaezono/example_alias」というファイルを more で閲覧して見て下さい (タブ補完利用！)。先程の'enw', 'cup' の他、幾つかのエイリアス登録が記載されているのが確認できます。このエイリアスファイルに対し、「**source** [エイリアスファイル名]」と入力すれば、記載登録されているエイリアスがシステムに反映されます。今の場合ならば、

```
% source ~/work/setupMaezono/example_alias
```

とする事で登録エイリアスがシステムに呑ませる事が出来ます。上記で実際に、'enw' や'cup' が有効に利用出来るようになった事を確認してください。ただ、このやり方では「エイリアスファイルが置いてある位置を毎度思い出す」という手間があります。通常、エイリアスファイルは、「普通は此の位置に、こういうファイル名で置く」という慣用法が決まっていますので、次にこれについて説明します。

ホームディレクトリに戻って「ls」と叩いて出力を確認した後、今度は

```
% ls -a
```

というオプション付きで叩いてみてください。.cache などといった「ドット (.) から始まる名称のファイルやディレクトリ」が大量に表示される事がわかります。これら「ドット (.) から始まる名称」のものを**不可視ファイル**といいます。ファイルの中でも「システムの設定などに関わるファイル」というのは、「一般的なファイル」(ユーザが作成した文書ファイル) と少し風合いが違うので、同じ「ファイルとして見えてしまう」のを避けるために、「特殊なオプションを附さないとデフォでは見えない」という風にしているのです[*20]。

alias ファイルも、この手のものと同列の不可視ファイルで、通常は「~/**.alias**」とし

[*20] 我々の日常生活でも、部屋にある冷蔵庫や椅子の他に、あまり見たくないもの (臭い気体の分子とか、足元がうっすら消えている人物など) まで見えてしまうと、色々と不都合が生じます。

て配置しますので、

```
% cd
% cp ~/work/setupMaezono/example_alias .alias
```

とします[*21]。再度「ls -a」を叩いて.alias が出来上がっていること、more で.alias の内容を確認し、正しく複写がされていることを確認します。そうしたら一旦「exit」と打ってターミナルを閉じてください。

再度ターミナルを立てたら、まずは'enw' や'cup' が無効になっていることを確認し、次に、「source ～/.alias」と打ち、エイリアスが通る ('enw' や'cup' が効くようになる) 事を確認して下さい。以降、このエイリアスファイルを使っていきます[*22]。新しく登録したいエイリアスがあれば、.alias に書いてある記載に倣って、次々と内容を育てていく事ができます。

エイリアスは勿論、個人で勝手に設定してもいいのですが、特定のグループやコミュニティ内でエイリアスを共有すると共同作業などを大きく加速する事ができます。筆者のグループで共用しているエイリアスは、筆者が元々、在籍していたケンブリッジのグループでタウラー先生[a]が使っていたものです。世界中から、若い時期に此のグループに在籍して「CASINO」というシミュレーション手法を学んだ人達が、その後、世界中に散って夫々のグループを主催し、さらに若い世代の仲間が再生産されていくわけですが、そうした人達が自分と同じエイリアスを使っていると、なかなか感慨深いものがあります。

[a] Mike D. Towler 博士

1.3.2　パイプとリダイレクト

Linux で便利な機能の一つに、「あるコマンドの出力を、別コマンドの入力として食わせる」という機能があります。「この機能がなくなってしまったら一気に魅力半減」といった「ノアの箱舟に載せる」チックな魅力のある機能です。

ホームディレクトリに移動して、ここで「du -h .cache/」とコマンドを叩いてみてください。画面がしばらく流れていくと思いますが、このコマンドは、「du [ディレクトリ名]」

[*21] 直前に同じ引数で source コマンドを叩いているので、ベタ打ちせずに上矢印キーで戻ってから Ctrl+A で前に戻り、Ctrl+D で source という文字列を消してから cp と打ち、Ctrl+E で行末に移動して.alias を追記するといったやり方が出来れば大したものです。

[*22] これだと毎度、ターミナルを立ち上げるたびに、ホームディレクトリにいることを確認して「source .alias」を実行しなければならないですが、もっと便利な利用の仕方を §1.3.3 で解説します。

として使うことで、当該ディレクトリ下に存在するファイルの容量を表示するコマンドです (「-h」は容量をキロやメガ単位で表示させるオプション) [*23]。不可視ディレクトリ「.cache」の内容はそこそこあるので、画面が速く「巻物で流れてしまう」(長い巻物の文書のように 1 画面で表示し切れず、人間が読み終わる前に先に先に自動スクロールで流れていってしまう) ので、最初に表示される部分を読み取るには、マウスを利用して延々とさかのぼっていかなければなりません。。ここで、「du -h .cache/ | more」と縦棒で繋げてコマンドを叩いてみてください。この場合、1 画面分表示されたら表示が停まります。あとは more の操作で、「エンターなら 1 行、スペースなら 1 画面分だけ」流れていきます。

「|」という縦棒を使って「[前半のコマンド]|[後半のコマンド]」という使い方をしましたが、これは「前半コマンドの出力を後半コマンドに引き渡す」という Linux の便利な使い方で、「前半コマンドを後半に**パイプ**」すると訓じます。今の例では、「du -h」で標準出力 (普通には画面のことです) に延々と流れ出る出力を、「more」の入力として食わせて、more コマンドで操作したという流れになります。

同じような「ノアの方舟に載る」的な使い勝手のよい機能として、リダイレクトというのがあります。「cd ～/work/setupMaezono/us」とディレクトリを移動して (タブ補完！)、ここで du と叩いてみてください。数行の表示が出ると思いますが、次に、「du > temp」と叩くと、temp というファイルが出来上がると思いますので、その内容を cat か more で確認してみます。先程、「du」と叩いて標準出力 (画面の事) に表示された内容が、temp というファイル内に書き込まれている事がわかります。「[コマンド] > [ファイル名]」とすることで、前段のコマンドで吐き出された内容が当該ファイルに書き込まれるという機能で、「[ファイル名] のファイルに**リダイレクト**する」という言い方をします。

このように、コンピュータの出力を手書きで書き取ったり、カットアンドペーストして記録を残す必要はなく、リダイレクトでファイルに保存できると言うことがわかります。とは言え、「必要な箇所だけ保存すればいい。リダイレクトでは、すべて保存されてしまうではないか」と思われるかもしれません。これについては、「grep/awk/sed」といった**テキスト処理**の便利なコマンドを本文中 §3.4 に紹介しますが、これらとパイプで組み合わせることで、恐ろしく便利に「ほしい箇所だけファイルに書き写す」という事が可能になります。これら機能を習得してしまうと、これまでエクセルを使っていたような処理について、有料のエクセルを使うのが馬鹿らしくなるくらい便利な生活が待っていることになります。

次に「echo 'student'」と叩いてみてください。「student」と返値されるかと思います

[*23] du は、システム管理をする際に利用するコマンドで、特に「du」の機能を初学者に教えようという意図はなく、パイプを使うのに手軽で最適なコマンドだったというだけの理由ですので安心してください。

が、この echo というコマンドは、「echo ’[文字列]’」とすると、その文字列が返値されるというコマンドです[*24]。これを「echo ’student’ > temp」とリダイレクトして、cat で temp の中身を見てみると、先程、temp に書き込まれた内容がすべて**上書き**されて「student」だけになってしまった事が確認できると思います。そうしたら、次に、「du >> temp」としてみます。これまでのリダイレクトが「>」だったのに対して、「>>」によるリダイレクトです。temp の内容を確認すると、「student」と書かれた行の次から du の出力が書き込まれているのが確認できます。つまり、「>」のリダイレクトは上書きしてしまうのに対して、「>>」のリダイレクトは「上書きせずに追記」として機能するというものです。

リダイレクトは「>」として出力に使う他に、「<」として入力に使うことも出来ます：まず、「bc」というコマンドを実行してみてください。カーソルの入力待ち状態になると思いますが、「3+5」と打ってエンターを押してください。8 が返値されると思いますが、「bc」はbasic calculator のニーモニックで、電卓コマンドとなっています。次に「5/3」を実行すると、答が 1 になってしまいます。そこで、「scale=3」と打ってから、再度 5/3 を実行すると (上矢印キーで辿れます)、1.666 と表示されます。「scale=3」が「小数点以下 3 桁で」という指示になっています。コマンドラインでの電卓機能は、カットアンドペーストで文字化けやフォント引き継ぎが生じないので、使い慣れると大変便利です。Ctrl+D を実行すると、電卓コマンドが終了され、通常のプロンプトが戻ってきます。

ここで、「echo ’3+5’ > temp」と実行し、3+5 が temp に書き出されている事を確認します。次に、「bc < temp」と実行して、8 が返値されることを確認してください。「[コマンド] < [ファイル名]」とすることで、「当該ファイルを当該コマンドの入力に喰わせる」ということが出来るのです。そうしたら今度は「echo ’scale=3’ > temp」として、次いで「echo ’5/3’ >> temp」とした後、temp の中身を確認して、「bc < temp」とします。所望の「1.666」が返値されることが確認されると思います。

次には、「bc < temp > out」として out の中身を確認してください。リダイレクトが初学だった読者にとっては、本設の説明前に「bc < temp > out」を見たら、<temp> という「カッコで囲まれた temp」が最初に目に入ってしまったものと思われますが、そうではなくて、「bc というコマンドに、temp という入力ファイルを食わせて、その出力を out ファイルに吐き出す」と読めるようになれば、初学者から一皮むけたということになります。

[*24] 初学の際には「こんなコマンド、何に使うんだ」と戸惑うコマンドですが、実は後述するスクリプトを用いる際には割と使い勝手のよいコマンドです。

1.3.3 ログイン後にエイリアスが自動で効くようにする

ところで §1.3.1 にて解説した「source .alias」の操作ですが、ターミナルを立ち上げるたびに、ホームディレクトリから毎度 source コマンドを実行するのは少し面倒です。ホームディレクトリ上に「.bashrc」というファイルが見つかります。その内容を cat で見てみると、

```
% cd
% cat .bashrc
    ...
    source ~/work/setupMaezono/bash_alias
```

と末尾に「bash_alias」というファイルを「source」で有効化するコマンドが書かれていることがわかります。Ubuntu の場合、ターミナルを立ち上げると「.bashrc」に記載されたコマンドが自動実行される規約になっているのです。この末尾の行は、この演習に先駆けて、本文 §2.1.4 で echo 文とリダイレクトを使って設定されたものですが、本付録章をここまで読了していれば、その内容を理解できるはずです。この末尾の行によって、ログインしたら、本書実習に便利なエイリアス群が自動的に有効化するよう設定されているのです。尚、.bashrc ファイルは alias 有効化以外にも、「ログインしたら自動実行して欲しいコマンド」をここに追記する形で有効活用されます。

尚、これまでもターミナルを立ち上げた時、bash_alias は有効化されていたわけですが、「source .alias を行わないとエイリアスが効かない」という説明がなされてきました。これは、例題に'enw', 'cup' といったエイリアスを使っていたからです。「source .alias」で有効化したのは

```
/Users/maezono/setupMaezono/example_alias
```

ですが、ターミナルを立ち上げると'.bashrc' によって自動的に立ち上がっていたのは、

```
/Users/maezono/setupMaezono/bash_alias
```

です。'enw', 'cup' は example_alias にしか含まれていないので、これらエイリアスは'source .alias' を行わないと有効化しなかったのです。example_alias は教育用の目的で利用しましたが、以降は、bash_alias を利用します。

以上で本書に必要最低限の Linux 教程は終了です。本文の §3 に戻り、実習を続けてください。