LINUX BASIC キャッチアップ

～理解できたかな？と疑問符が付く人へ～

Ver. 1.0

リナックスアカデミー矢越昭仁

2012/04/28

Linux BASIC の途中でつまずきやすい点を、より詳しく再度解説します。何のために学んでいるのか、 わからない時にすべきこと、ステップアップするためには、といった視点で解説します。

目次

はじめに

IT 業界において技術的な知識とは別に最も重要なのは「飽くなき探究心＝興味」だといわれています。この IT 特別講座では皆さんの知識・技術の向上を促し、「飽くなき探究心＝興味」を満足させる講座を提供することを目的としています。講座中は遠慮なく質問し、より理解を深めるとともに、新たな疑問は次の講座の開催要望として意見をください。

表記について

この資料では以下の表記としています。

・フォント

コンピュータの操作および設定ファイルはクーリエフォント(タイプライター風)を用います。

```
search t123006.la.net
nameserver 10.20.123.6
```

・プロンプト

コマンド入力例がある場合は、先頭はプロンプト($または#)で始めます。

$ は一般ユーザでの操作、#はルートユーザでの操作を表します。なおユーザ切り替え(su)は省略しています。

・強調(ボールド)

コマンド入力では、キーボードから入力する場合を、設定ファイルの場合は修正箇所など特に強調したい場合に**ボールド**を使います。

```
$ date
Mon Mar 5 12:32:41 JST 2012
```

・凡その作業時間

凡その作業時間とは、過去に同様の作業を経験した人が再度実行した場合にかかる時間を想定しています。つまり事前調査や試行錯誤の時間を含まない作業時間を指します。

オンラインバックアップ

矢越が実施した IT 特別講座の資料(補足資料、例題等含む)は、以下の URL にて掲載しています。この URL はリナックスアカデミー会員限定となっていますので、それ以外への再配布・再掲載は遠慮ください。

http://ycos.sakura.ne.jp/LA

また講座・資料への質問、要望は下記までメールをお願いします。

mailto:ycos001@yahoo.co.jp

キー操作

Linux は CUI(Character based User Interface, Command line ～)を基本としており、どんな操作を行うにしてもコマンド入力が必要です。特に実習ではキー操作でまごつく事がないよう、しっかりとした基礎を身につける必要があります。
タイプでまごついて、肝心の講義内容を聞き逃す、タイプに集中する余りシステムが表示したメッセージが目に入らないといった事は、実習において致命傷です。

削除キー

入力している途中で1文字削除するには、[Back Space]や[Delete]を用います。この違いを明確に理解していない人も意外といるので、ここで整理しておきます。

1文字削除する	
[Back Space]	カーソルが点滅している左側を1文字削除し詰めていきます。 ^H でも代用できます。
[Delete]	カーソルが点滅している直下の文字を削除します。
まとめて削除する	
^U	カーソルの左から行の頭までを削除します。 パスワード入力中に打ち間違った時などに利用すると便利です。
^L	画面を消去します。 clear(1)と同じですが、入力途中の文字列は残したままです。

例) [Back Space]と[Delete]の違い
以下のように、d にカーソルが位置づいている時に、

```
$ abcdefg
```

[Back Space]を押せば、カーソルの**左の１文字**が削除され、カーソル以降は左に詰められます。

```
$ abdefg
```

続けて、^L を押せば、カーソルから左の文字は全て削除されます。

```
$ efg
```

再び、最初の状態から、

```
$ abcdefg
```

[Back Space]を押せば、カーソル**下の１文字**が削除され、カーソル以降は左に詰められます。

```
$ abcefg
```

コマンド補完

長いファイル名を一度にタイプミスなく入力するのは至難の業です。特に大文字小文字が入り混じった、紛らわしい名前のファイルのどは、途中で[TAB]を使いファイル名補完を活用するとよいでしょう。
実習などでは、必ずファイル名置換を使いながら、ファイル名を入力すべきです。

例) ファイル名を補完する
例えば、/etc/resolv.confを表示する場合、表示コマンド cat に続け、/etc/res までタイプし、[TAB]により残りを補完します。

```
$ cat /etc/res[TAB]
$ cat /etc/resolv.conf
```

途中まで入力したファイル名で始まるファイルが複数存在する場合は、最初の[TAB]で Beep(ピッというベル音)が鳴ります。その場合は、再度[TAB]を押すと候補一覧が表示されます。

```
$ cat /etc/re[TAB] ← ピッ
(再度、[TAB])
readahead./   redhat-lsb/          resolvconf
reader.conf   redhat-release       resolv.conf.predhclient
(d を追加し、再び[TAB])
$ cat /etc/redhat- ←今度は、redhat-で始まるものが複数ある
(再度、[TAB])
readhat-lsb   readhat-release
```

(r を追加し、[Enter])

一覧が表示された場合は、必要なファイルが補完されるまで少しずつ文字を追加し、[TAB]を入力することで、最終的に必要なファイル名全てを補完する事ができます。

コマンド履歴

過去に実行したコマンドは history(1)によって、参照することができます。
複雑な手順を伴う実習で、うまく動かない場合は、順序を間違っている可能性があります。操作対象となっているファイル名や指示コマンドが似ていて、間違い易いといった事もよくあります。

```
[student@cent570 temp]$ history
  216  vi a.lis
  217  cd ../Openlab2/
  218  ls -l
  219  ls -l > 00Readme.txt
  220  cat ydup
  221  grep sed ydup
  221  sed -e '/^#H/s/^..//p' -e d ydup
  222  wc -l *
  223  grep -w shift *
  224  tar czvf Openlab2.tar.gz Openlab2
  225  rm Openlab2.tar
  226  mv Openlab2.tar.gz   /win/temp/
  227  su –
```

!を使ってコマンドを再実行する事ができます。
例)

!コマンド	実行結果
!!	直前のコマンド、この場合は 227 行目の su -
!222	222 番目のコマンド、この場合は wc –l
!grep	遡って grep で始まる最初のコマンド。この場合は 223 行目の grep –w shift *

カーソルキーなどを使い、過去の履歴を参照・修正し再実行する事もできます。コマンドを修正し、必要な内容になった時点で[Enter]を押せば、それが実行されます。
この時、カーソルはコマンド行のどこにあっても構いません。行の途中であっても[Enter]が入力されれば、表示されている1行分全てを入力したものとして、実行されます。

コマンド行編集時のキー操作

キー	動作
↑または^P	コマンド履歴を遡る(上方向へ移動)
↓または^N	コマンド履歴を進む(下方向へ移動)
←または^B	1文字左へ(行頭方向)へ移動
→または^F	1文字右へ(末尾方向)へ移動
^A	行頭へ移動
^E	行末へ移動
^W	行頭方向へ、1単語削除し移動
[ESC]後、B	1単語(空白、特殊文字で区切られた文字並び)分右へ
[ESC]後、F	1単語(空白、特殊文字で区切られた文字並び)分左へ

詳しくは bash(1)のマニュアル、readline の Commands for Moving を参照のこと。

コマンドと引数

Linuxは「寡黙なOS」と呼ばれ、エラーがない限りメッセージは表示しません。ファイルを削除するといった、危険な行為でもエラーでなければ寡黙に実行します。
つまり的確にコマンドを入力し、確実に実行した事を確認しながら操作する事が必要です。

コマンドライン

プロンプト(入力促進記号、$ や＃)の直後から、[Enter]を押すまでの間に入力された文字列をコマンドライン(コマンド行、コマンド列とも)と呼びます。シェルはプロンプトを表示し、入力を待ちます。
シェルはコマンドラインを以下の手順で処理します。

1. 字句解析(Lexical Analysis)
   コマンド列を単語に分割します。分割は予め決められた「区切り文字」により行われ、分割された単語をトークン(token)と呼びます。
   Bash の区切り文字は、空白、タブ、[Enter]、|、&、;、()、<、>及び＃になります。中には2つ連ねる事で意味が変わる文字もあります。
   そして、最初のトークンをコマンド(command)、続くトークンを引数(ひきすう、argument)と呼びます。Unix/Linux ではさらに、引数の中でもハイフンで始まるものをオプションと呼びます。

2. 構文解析(Syntactic Analysis)
   続いて各トークンの意味を解析します。最初の単語に基づき必要な数の引数があるかどうか、メタキャラクタであれば対応するファイル名に置換、$を含む単語であれば変数として、その値への置き換えなども行います。
   例えば引数に「a*」が指定されていれば、a で始まるファイル名群に置き換えます。不等号＞があれば、それに続く引数(出力ファイル名)があるかどうかの確認をします。この様な特殊文字を他の値に変換する事を「コマンド列を**評価する**(evaluation)」と言います。

3. 実行(Execution)
   最後にコマンド列から切出したコマンドを調べ、Bash 自らの動作を指示しているものであれば、そのまま実行(これを Bash 組み込みコマンドと呼び、for, while, if, echo などがあります)。
   Bash がそのまま実行できない、知らないコマンドであれば環境変数 PATH で指定されたディレクトリに同じ名前の実行可能ファイルがないかを検索します。
   実行ファイルが見つかれば、そのプログラムへ評価が終わった引数を引き渡し、実行します。その後プログラムが終了するまで停止し、終了したら再びプロンプトを表示しコマンド待ちになります。

例)
以下のコマンドを実行し、なぜそうなるか考えてみてください。

```
$ echo *
$ echo a          b    c    d
$ echo “a          b     c    d”
$ echo a b c # test
```

特殊文字

Bash が評価する(何かと置き換える)特別な文字を、特殊文字(Special Character、Meta Character)と呼び、以下のような種類があります。

文字(よみ)	意味
＊(アスタリスク)	0文字以上のファイル名、ファイルがない場合は＊
？(クエスチョン、ぎもんふ)	1文字のファイル名、ファイルがない場合は？
～(チルダ、Shift+[へ])	ホームディレクトリパス名
＼(バックスラッシュ、端末によっては¥)	続く特殊文字を評価しない。文字として処理
$(ダラー)	続く文字列を変数名として、その値に変換
！(エクスクラメーション、バン、びっくり)	続く文字列で始まるコマンド実行履歴に変換
＃(シャープ、ハッシュ、いげた)	以降コメントとして無視

例)

```
$ ls  -l  /etc/*/*/httpd.conf
-rw-r--r-- 1 root root 33726 Jan 17 18:47 /etc/httpd/conf/httpd.conf
```

コマンドを理解する

コマンドを覚えるには、実際に操作をする事が重要です。「習うより慣れろ」的ですが、覚えるのはこれが確実です。仮想システム(VMware Player, Oralce VirtualBox)を使えば、Windows PC 上でも Linux を動作させる事が出来ます。また制限は多いですが、Cygwin(エミュレータ)でも基本的なファイル操作コマンドは利用できます。

「Windows PC での仮想 Linux 環境構築演習」
http://ycos.sakura.ne.jp/LA/Special/HomeLinux3-1.1.pdf

基本的なコマンドは短く覚えにくいとよく言われますが、英文のマニュアルを参照すれば謂(いわれ)れが分かります。

```
$ (export LANG=C;man ls)
```

cmd	いわれ	意味			
ls	list	ファイルの一覧	mkdir	make directory	ディレクトリの作成
rm	remove	ファイルの削除	less	opposite of more	more を改良したので
mv	move	ファイルの移動	chmod	change mode	モードの変更
cp	copy	ファイルの複写	chown	change owner	所有者変更
vi	visual	スクリーン(画面)	pwd	print working directory	作業ディレクトリの表示
cat	con**cat**enate	連結	cd	Change directory	ディレクトリ移動
su	set user	UID 切り替え	umask	user file creation mask	モード初期値のユーザ設定

またコマンドを覚えるとは、コマンド名だけでなくよく使うオプションも一緒に覚える必要があります。一律の規則はありませんが、多くのコマンドでは以下のようなオプションがあります。

opt	いわれ	意味
-l	long	詳細な情報
-n	number	ホスト名やポート名を番号で
-r	recursion	ディレクトリ階層すべて(再帰)
-v	verbose	途中経過を表示(おしゃべり)
-f	force	強制的に実行、確認無視

コマンドを覚えるコツ

1. コマンドノートを作る
   手帳サイズのノートに、1コマンド1葉で分かりやすいタイトルをつけ、良く使うオプションや実行例をメモします。特に単純なコマンド実行例だけでなく、パイプ(|)を使った例などを記録するようにしましょう。現場でも役に立ちます。
   タイトルをつけるときは簡潔に、「○○を××する」といった形式にすると引き出しやすくなります。一つのコマンドで多くの機能があり、この様なタイトルにできない場合は、用紙を分けて書きます。

2. 沢山こなす
   とにかく沢山のコマンドをこなすために、知らないコマンドでもマニュアルを参照して実行します。参照系(xxの一覧、xxの表示といった類)であれば、多少失敗しても問題ありません。一般ユーザで、どんどんコマンドを実行しましょう。沢山のコマンドに触れれば、感覚的にオプションが身に付きます。
   コマンドの一覧は、マニュアルの格納先を参照するとよいでしょう。
   ```
   $ ls  /usr/share/man/ja/man1      ←日本語 manpage(1)
   $ ls  /usr/share/man/man1         ←英語 manpage(1)
   ```
   因みに CentOS 5.7 では、第 1 章(一般ユーザ向けコマンド)のマニュアル登録数は 1,250 あります。

設定ファイルの修正

vi 再び

vi エディタにはなれましたか？Linux のシステム管理では必ず vi(vim)を使う事になるので、しっかり押さえておきたいツールです。
基本的な操作にいては、すでに Linux BASIC で扱っていますが、vim 自身にも入門用教材があります。vimtutorコマンドは入門用コマンドで、実際にvim が起動され、練習問題をこなすことで vim の操作が身に付きます。

```
$ vimtutor
```

```
===============================================================
=      ＶＩＭ 教 本 (チュートリアル) へ よ う こ そ          -      Version 1.7 =
===============================================================
Vim は、このチュートリアルで説明するには多すぎる程のコマンドを備えた非常
に強力なエディターです。このチュートリアルは、あなたが Vim を万能エディ
ターとして使いこなせるようになるのに十分なコマンドについて説明をするよう
なっています。
チュートリアルを完了するのに必要な時間は、覚えたコマンドを試すのにどれだ
け時間を使うのかにもよりますが、およそ 25 から 30 分です。
ATTENTION:
以下の練習用コマンドにはこの文章を変更するものもあります。練習を始める前
にコピーを作成しましょう("vimtutor"したならば、既にコピーされています)。
このチュートリアルが、使うことで覚えられる仕組みになっていることを、心し
ておかなければなりません。正しく学習するにはコマンドを実際に試さなければ
ならないのです。文章を読んだだけならば、きっと忘れてしまいます!。
さぁ、Caps ロック(Shift-Lock)キーが押されていないことを確認した後、画面に
レッスン 1.1 が全部表示されるところまで、j キーを押してカーソルを移動しま
しょう。
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
レッスン 1.1:  カーソルの移動
**  カーソルを移動するには、示される様に h,j,k,l を押します  **
```

vimtutor は、パッケージ vim-enhanced に含まれています。

転ばぬ先の修正

各種設定ファイルを修正する際には、以下の点に留意しましょう。

・別名保存する
設定ファイルを大きく変更する場合は、もともとのファイルをコピーし最悪の場合でもそのファイルを使って復帰できるようにします。

例えば

```
# cp /etc/hosts   /etc/hosts.org
```

というように、元のファイルを別名でコピーしておいてから、/etc/hosts を修正します。
よく変更するファイルであれば、世代管理を行いいくつもバージョンを用意しておくとよいでしょう。

```
# cp /etc/hosts   /etc/hosts.20120406
```

vim では編集中のファイル名は % で表す事ができます。もし vi でファイルを開いて修正を始めてしまった場合は、保存する前に(ファイルを変更してしまう前に)次の ex コマンドで、別名保存ができます。

```
:! cp   %    %.org
```

ex コマンドの ！ は、以降[Enter]までをシェルとして実行します。

・コメントの活用
実際の現場では、複数のシステム管理者が一つの設定ファイルを修正する場合があります。また設定ファイルにかぎらず、ソフトウエアに変更を行う場合は必ずその履歴を残します。いつ、誰が、何の目的で、

どこを、どのように修正したかを管理しています。特に基幹業務に係るような場合は、いきなり変更するのではなく、変更して大丈夫なのか事前によく調査し、テスト環境で検証し、その時と全く同じ手順で実システム(本番系、本番環境、Production システム)を修正します。
修正箇所をより分かりやすくするため、コメントを使い記録しておきます。
例)

```
# 2012/04/06 Yakoshi, Change StartServers 8 ---> 4
# StartServers   8
StartServers  4
        :
# 2012/03/30 Yakoshi, UserDir setting change +++++++++++++++
<IfModule  mod_userdir.c>
        # UserDir  disable
        UserDir  public_html
</IfModule>
# 2012/03/30 Yakoshi, UserDir setting change ---------------
```

実際には、何らかのトラブルにより設定を修正することがほとんどです。トラブルは連番管理され、修正箇所にもその番号が付与するのが一般的です。

変更の確認

また修正前と修正後を比較するには、diff コマンドを用います。

```
[root@cent570 temp]# cat -n h1
     1   # Do not remove the following line (/etc/hosts), or various programs
     2   # that require network functionality will fail.
     3   127.0.0.1               localhost.localdomain localhost myhost
     4   ::1             localhost6.localdomain6 localhost6
     5   192.168.182.128 h128.s182.la.net h128
[root@cent570 temp]# cat -n h2
     1   # Do not remove the following line, or various programs
     2   # that require network functionality will fail.
     3   127.0.0.1               localhost.localdomain localhost myhost
     4   192.168.182.128 h128.s182.la.net h128
     5   192.168.182.130 h130.s182.la.net h130

[root@cent570 temp]# diff h1 h2
1c1
< # Do not remove the following line(/etc/hosts), or various programs
---
> # Do not remove the following line, or various programs
4d3
< ::1           localhost6.localdomain6 localhost6
5a5
> 192.168.182.130      h130.s182.la.net h130
```

diff は 2 つのファイルで違う箇所を検知し、変更の詳細を表示します。

[行数From][変更種別][行 To]
< 引数、右のファイルを左のファイルにする為に必要な変更内容

> 引数、左のファイルを右のファイルにする為に必要な変更内容

変更種別は c:置き換え、a:追加、d:削除となり、行単位での指摘となります。
上記の例では、

- 1 行目を変更(1c1)
  h1 では(/etc/hosts)という内容がありましたが、h2 ではそれがないため、この行が変更された事を表します。
- 4 行目削除(4d3)
  h1 の 4 行目を削除し、h2 の 3 行目にした事を表します。
- 1 行削除(5a5)
  h1 の 5 行目に追加。先の削除があるので、結果として h2 の 5 行目として 1 行追加

-e オプション(edit)を使えば、左のファイルを右のファイルにする為に必要な ed(ex)コマンドのスクリプトを作成します。

```
[root@cent570 temp]# diff -e h1 h2
5a
192.168.182.130 h130.s182.la.net h130
.
4d
1c
# Do not remove the following line, or various programs
```

-c オプション(context)は変更箇所が分かりやすいように、その前後を表示してくれます。慣れないうちは、-c オプションが便利でしょう。

```
[root@cent570 temp]# diff -c h1 h2
*** h1 2012-04-10 07:58:32.516903100 +0900
--- h2 2012-04-10 07:58:32.521904100 +0900
***************
*** 1,5 ****
! # Do not remove the following line(/etc/hosts), or various programs
  # that require network functionality will fail.
  127.0.0.1           localhost.localdomain localhost myhost
- ::1         localhost6.localdomain6 localhost6
  192.168.182.128     h128.s182.la.net h128
--- 1,5 ----
! # Do not remove the following line, or various programs
  # that require network functionality will fail.
  127.0.0.1           localhost.localdomain localhost myhost
  192.168.182.128     h128.s182.la.net h128
+ 192.168.182.130     h130.s182.la.net h130
```

2 つのファイルを比較し、他方とどこが違うかを表示します。
行頭に変更種別があり、!:置き換え、+:追加、-:削除となります。

また、パッケージであればインストール後に修正されたファイルを rpm(8)の-V オプションで調べることができます。

```
[root@cent570 temp]# rpm  -V  httpd
S.5....T  c /etc/httpd/conf/httpd.conf
```

ファイル名の前にある記号の羅列は、インストール時と何が変わったかを表しています。ドット(.)は変化がない事を表しています。

- S　ファイルの大きさ(Size)
- M　モード、パーミッション(Mode)
- 5　MD5 チェックサム
- D　デバイスファイルのメジャー、マイナー番号(Device)
- L　リンク先(Link)
- U　所有ユーザ(User ownership)
- G　所有グループ(Group ownership)
- T　タイムスタンプ（最新更新日時、Time）

パーミッション

Linux のファイル（情報）保護は、ファイルのパーミッション（Permission、許可の意味）によってのみ実現されます[1]。つまり一旦 Linux にログインした状態で、ユーザやシステムの情報を守るためには、この属性をいかに制御できるかにかかっています。
非常に重要な機能ですので、しっかりと理解しておく必要があります。

所有者と許可

Linux のファイルには、所有者のユーザIDとグループID情報が含まれます。これはファイル作成時に付与され、ファイルを作ったユーザのUIDとプライマリーGIDになります。

- ファイルの所有ユーザは、root 以外は変える事はできません。
  （勝手に人の持ち物を、自分の物に変える事はできません。）
- ファイルの作者は、自分が所属するグループへ所有グループを変更する事はできます。
  root は自分の所属に関係なく、所有グループを変える事ができます。

また、以下に解説するパーミッションは書き込み権があっても作者以外は変更できません。（root 除く）

さらにファイルには、3 つの操作（r:参照 read、w:変更・上書き write、x:実行 execution）について制限をかける事ができます。許可がない行為は作者であっても行う事ができません。この許可のことをパーミッションと呼びます。
操作はファイル作者からみて、次の 3 種類のユーザに分類されます。u:同一ユーザ（作者本人、user）、g:作者が所属するグループのメンバー(group)、o:全く知らないユーザ(other)。
ファイルを操作しようとする人が、そのファイル作者から見て、どの種類にあたるかで、先の許可が決まります。また、許可があるかどうかは、u>g>o の順に評価されます。

1. まずファイルの作者と、操作する人が同じユーザかどうかを判定、同じであれば権限があるかどうかを判断。
   許可がなければ、次へ（以下同様）
2. ファイルのグループと、操作をする人が同じプライマリグループかどうかを判断、同じであれば権限を判定。
3. ファイルのグループと、操作をする人が所属するセカンダリグループが同じかどうかを判断、同じであれば権限を判定。
4. 最後に、権限を判定し第 3 者で行うことができなければ、最終的にエラーとして終了。

[1] 実際にはセキュリティを拡張した SELinux や、アプリケーションごとのセキュリティがあります。

これから

Linux BASIC で得た知識は、LPIC Level 1 試験相当になります。詳しくいうと 101 試験の殆ど、102 試験は少し勉強することで Level 1 を取得する準備が整ったわけです。

ＬＰＩＣ試験

LPIC 試験はレベル1～3に分かれています。これは経済産業省が示すITスキルのガイドライン、ITSS (IT Skill Set)の習熟度レベルと合致しています。

レベル	内容
1	初心者・入門者 情報技術に携わる者に最低限必要な基礎知識を有する。
2	若手社員レベル 上位者の指導の下に、要求された作業を担当できる。
3	中堅社員レベル 要求された作業を全て独力で遂行できる。
4	プロフェッショナル スキルの専門分野が確立し、独力で業務上の課題の発見と解決をリードできる。
5	社内トップクラス 社内においてテクノロジやメソドロジ、ビジネスを創造しリードできる。
6	国内トップクラス 全国的にテクノロジやメソドロジ、ビジネスを創造しリードできる。
7	グローバルクラス 世界的にテクノロジやメソドロジ、ビジネスを創造しリードできる。

（参考）IPA ホームページ　http://www.ipa.go.jp/jinzai/itss

つまりレベル1は「Linuxとは何か知っています」、レベル 2 で「Linuxの作業ができます」といった習熟度会いです。
またレベル 1 は知識だけで取得できますが、レベル 2 は実務経験を問われる「トラブルシューティング」の比重が多くなります。実機を使った操作・研究がなければ、取得は難しいでしょう。

Linux BASIC 受講後の技術を向上するステップとしては

1. **101 試験合格**
   まずは一通り復習して 101 試験に合格する事がはじめの一歩です。添付資料にもありますが、Linux BASIC の範囲を完全に理解していれば 101 試験に合格可能です。
2. **自習**
   Linux BASIC だけでは不足する部分が 102 試験にはあります。しかし 101 試験の内容を自分の物にしていれば、学習書や問題集を理解し独学で 102 試験を突破できる実力がついています。
   試験に備え、少し勉強するとよいでしょう。
3. **102 試験合格**
   102 試験に合格すれば、Level 1 の認定が受けられます。Linux BASIC 受講後、復習の中で受験勉強し、101/102 両方の試験を一度に受ける人もいます。
   LPI の認定は世界共通ですから、ITスキルは世界で認められた事になります。
4. **LPIC-Level 2 に挑戦**
   次のステップは Level2 です。レベル 2 では、ネットワークサーバ構築とトラブルシューティングに比重が移ります。独学では環境の整備に苦労するかもしれません。（サーバとクライアントが必要）
   またトラブルシューティングは実機を操作し「体で覚える」的な要素も多いため、実験できる環境の整備は必須です。
   LAの Linux MASTER は Level 2 相当の知識を提供していますが、合格のためには経験を積む必要があります。

Level 3 は、なかなかエンジニア以外の人が取得するのは難しい内容となります。毎日、Linux と格闘するぐらいの経験が必要です。（といっても、実際に直接仕事に携わらない合格者もいます）
せっかく Linux Academy で学ぶわけですから、Level 1 は必ず合格したい、してほしいところです。就職・転職で差別化するのであれば、Level 2 も欲しいところです。

ＬＡコースと試験主題

101 試験：Linux の基本操作

ID	主題	W	参照
101.1	ハードウェア設定の決定と構成	2	(PC、L1)
101.2	システムのブート	3	1章
101.3	ランレベルの変更とシステムのシャットダウンまたはリブート	3	1章、10 章
102.1	ハードディスクのレイアウト設計	2	10 章
102.2	ブートマネージャのインストール	2	10 章、(L1)
102.3	共有ライブラリを管理する	1	(LM)
102.4	Debian パッケージ管理を使用する	3	(8 章)
102.5	RPM および YUM パッケージ管理を使用する	3	8 章
103.1	コマンドラインで操作する	4	2 章
103.2	フィルタを使ってテキストストリームを処理する	3	7 章、(L1)
103.3	**基本的なファイル管理を行う**	**4**	2 章
103.4	**ストリーム、パイプ、リダイレクトを使う**	**4**	7 章
103.5	**プロセスを生成、監視、終了する**	**4**	6 章
103.6	プロセスの実行優先度を変更する	2	(L1)
103.7	正規表現を使用してテキストファイルを検索する	2	7 章、(L1)
103.8	vi を使って基本的なファイル編集を行う	3	5 章
104.1	パーティションとファイルシステムの作成	2	(L1)
104.2	ファイルシステムの整合性を保持する	2	(L1)
104.3	ファイルシステムのマウントとアンマウントをコントロールする	3	8 章
104.4	ディスククォータを管理する	1	(L1)
104.5	ファイルのパーミッションと所有者を管理する	3	3 章
104.6	ハードリンクとシンボリックリンクを作成・変更する	2	2 章
104.7	システムファイルを見つける、適切な位置にファイルを配置する	2	2 章、8 章

102 試験：初歩的なシステム管理（スタンドアローン環境）

ID	主題	W	参照
105.1	**シェル環境のカスタマイズと使用**	**4**	13 章、付録
105.2	**簡単なスクリプトをカスタマイズまたは作成する**	**4**	13 章、付録
105.3	SQL データ管理	2	(L1)
106.1	X11 のインストールと設定	2	(L1)
106.2	ディスプレイマネージャの設定	2	(L1)
106.3	アクセシビリティ	1	(L1)
107.1	**ユーザアカウント、グループアカウント、および関連するシステムファイルを管理する**	**5**	2 章、3 章
107.2	**ジョブスケジューリングによるシステム管理業務の自動化**	**4**	(LM)
107.3	ローカライゼーションと国際化	3	(L1、LM)
108.1	システム時刻を維持する	3	(L1、LM)
108.2	システムのログ	2	11 章、(LM)
108.3	メール転送エージェント(MTA)の基本	3	(LM)
108.4	プリンターと印刷を管理する	2	(L1)
109.1	**インターネットプロトコルの基礎**	**4**	9 章、(NW)
109.2	**基本的なネットワーク構成**	**4**	9 章
109.3	**基本的なネットワークの問題解決**	**4**	(L1、LM)
109.4	クライアント側の DNS 設定	2	14 章
110.1	セキュリティ管理業務を実施する	3	(LM)
110.2	ホストのセキュリティ設定	3	(LM)
110.3	暗号化によるデータの保護	3	(LM)

重:重要度。参照は Linux BASIC での登場箇所、()は Linux BASIC 本編以外のコース。
(PC):コンピュータ入門、(NW):ネットワーク入門、(L1):LPIC-1 対策、(LM):Linux Master