LinuC レベル 1 Version10.0
技術解説無料セミナー
2020/10/31 開催

主題　「シェルスクリプト/ジョブスケジューリング」
（1.06.2 / 1.08.2）

セミナー担当者

河原木忠司（KAWARAGI Tadashi）

LPI-JAPAN

Agenda

#LinuC学習中

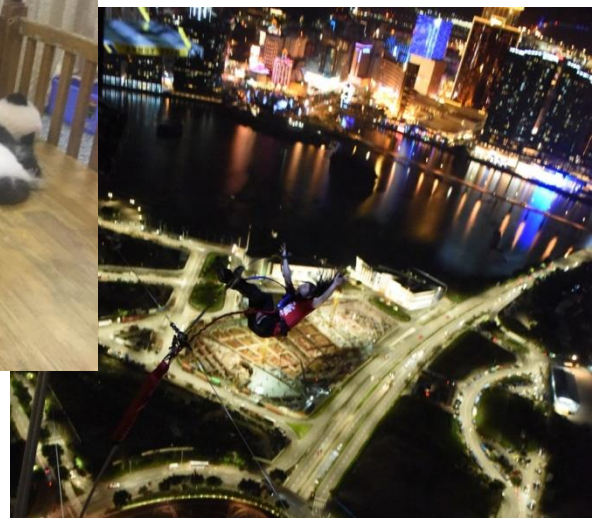
- 講師プロフィール、LinuC ver.10のご紹介
- 今回の主題についての概要

講師プロフィール

■河原木忠司（かわらぎただし）

- 20数年ほど、講師/エンジニアとして活動しております。
- 最近は講師、研修コンサルティング、執筆業に従事させていただいております。
- サーバーインフラ系のコース、セキュリティ系のコースを中心に担当させていただいております。
- 好きなもの
  - パンダ
  - 妻
  - うずらの卵
  - 音楽

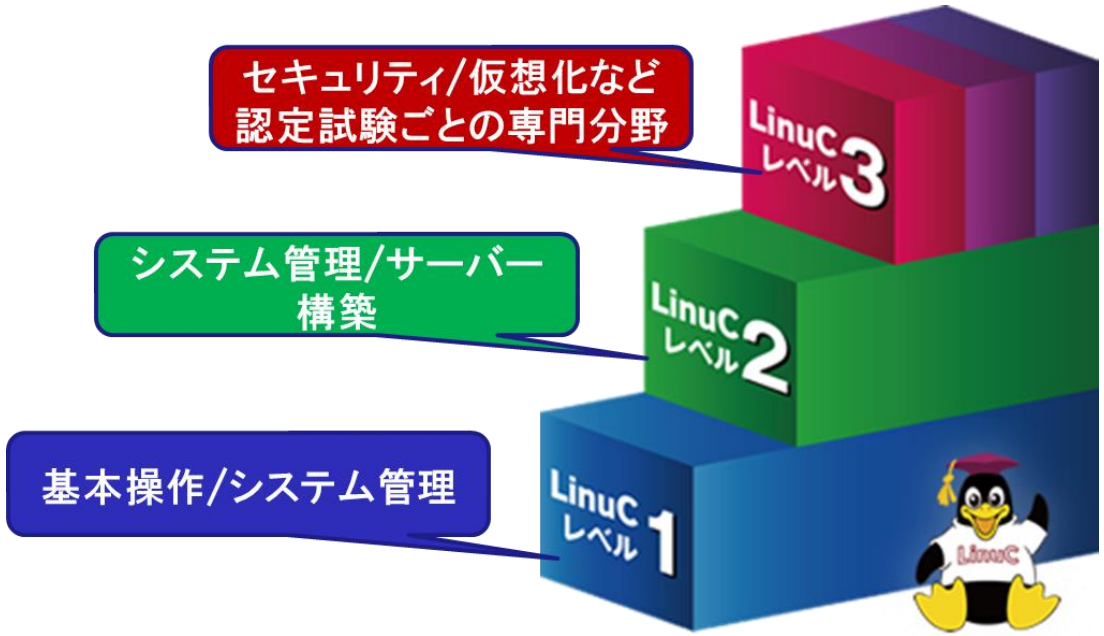
「最短突破 LinuCレベル1 合格教本 ver.10対応」（技術評論社）
好評発売中です。
※特典販売については最後にご説明いたします



LinuCとは

クラウド時代の即戦力エンジニアであることを証明するLinux技術者認定資格

- 3段階のレベルで構成
  - Lv.1～3で構成（Lv.1が基礎ベース）
  - Lv.1と2は2試験合格で認定
- 試験形式
  - CBT（Computer Based Testing）が基本
  - ペーパーベースの試験もあり
- 合格基準
  - 合格点は非公開。目安として65～75%程度の正解率で合格。
- 詳しくは・・・
  https://linuc.org/faq/



LinuC試験の出題形式

■選択問題
- 択一選択
- 複数選択

■一部入力問題あり

- コマンドをしっかり「覚える」必要がある
  - 「暗記」はつらいので、学習環境を用意して、入力しながら自然と「覚える」
  - ↓
  - 実践的なスキルの習得にもつながる

- 出題イメージ

- ファイルの末尾を表示するコマンドの名前を入力しなさい

_____

LinuCレベル１／レベル２ Version10.0とは

#LinuC学習中

✓現場で「今」求められている新しい技術要素に対応

- オンプレミス／仮想化を問わず様々な環境下でのサーバー構築
- 他社とのコラボレーションの前提となるオープンソースへの理解
- システムの多様化に対応できるアーキテクチャへの知見

✓全面的に見直した身につけておくべき技術範囲（＝出題範囲）

今となっては使わない技術やコマンドの削除、アップデートなど

✓Linuxの範疇だけにとどまらない認定領域

セキュリティや監視など、ITエンジニアであれば必須要件もカバー

■Version10.0と従来の出題範囲の比較

https://linuc.org/linuc1/range/
2021年3月末まではver.4と共に運用

	テーマ	Version 10.0	従来
LinuC-1	仮想技術	・仮想マシン／コンテナの概念 ・クラウドセキュリティの基礎	← （Version10.0で新設）
	オープンソースの文化	・オープンソースの定義や特徴 ・コミュニティやエコシステムへの貢献	← （Version10.0で新設）
	その他	→ （Version10.0で削除）	アクセシビリティ、ディスククォータ、プリンタの管理、SQLデータ管理、他
LinuC-2	仮想化技術	・仮想マシンの実行と管理（KVM） ・コンテナの仕組みとDockerの導入	← （Version10.0で新設）
	システムアーキテクチャ	・クラウドサービス上のシステム構成 ・高可用システム、スケーラビリティ、他	← （Version10.0で新設）
	その他	・統合監視ツール（zabbix） ・自動化ツール（Ansible）	← （Version10.0で出題範囲に含む）
		→ （Version10.0で削除）	RAID、記憶装置へのアクセス方、FTPサーバーの保護、他

今回のセミナーの主題と出題範囲との関連

#LinuC学習中

101試験

- 1.01：Linuxのインストールと仮想マシン・コンテナの利用
- 1.02：ファイル・ディレクトリの操作と管理
- 1.03：GNUとUnixのコマンド
- 1.04：リポジトリとパッケージ管理
- 1.05：ハードウェア、ディスク、パーティション、ファイルシステム

102試験

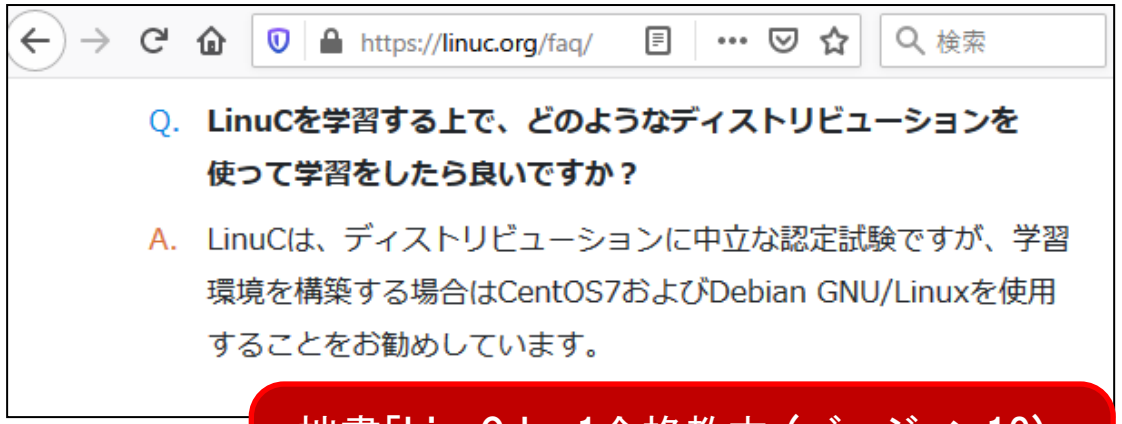
- 1.06：シェルおよびスクリプト
  - 1.06.1シェル環境のカスタマイズ
  - 1.06.2シェルスクリプト
- 1.07：ネットワークの基礎
- 1.08：システム管理
  - 1.08.1アカウント管理
  - 1.08.2ジョブスケジューリング
  - 1.08.3ローカライゼーションと国際化
- 1.09：重要なシステムサービス
- 1.10：セキュリティ
- 1.11：オープンソースの文化

これらの範囲から抜粋して、
解説をします

デモ環境について

■以下の環境を構成して、デモ操作をしながら解説します。

- CentOS7
- Debian10



拙書「LinuC Lv.1合格教本（バージョン10）」の付録DVDに含まれている仮想マシン環境を利用します。

■利用するユーザー

ユーザー名	ログインシェル
root	/bin/bash
testuser	/bin/bash
dashuser	/bin/dash

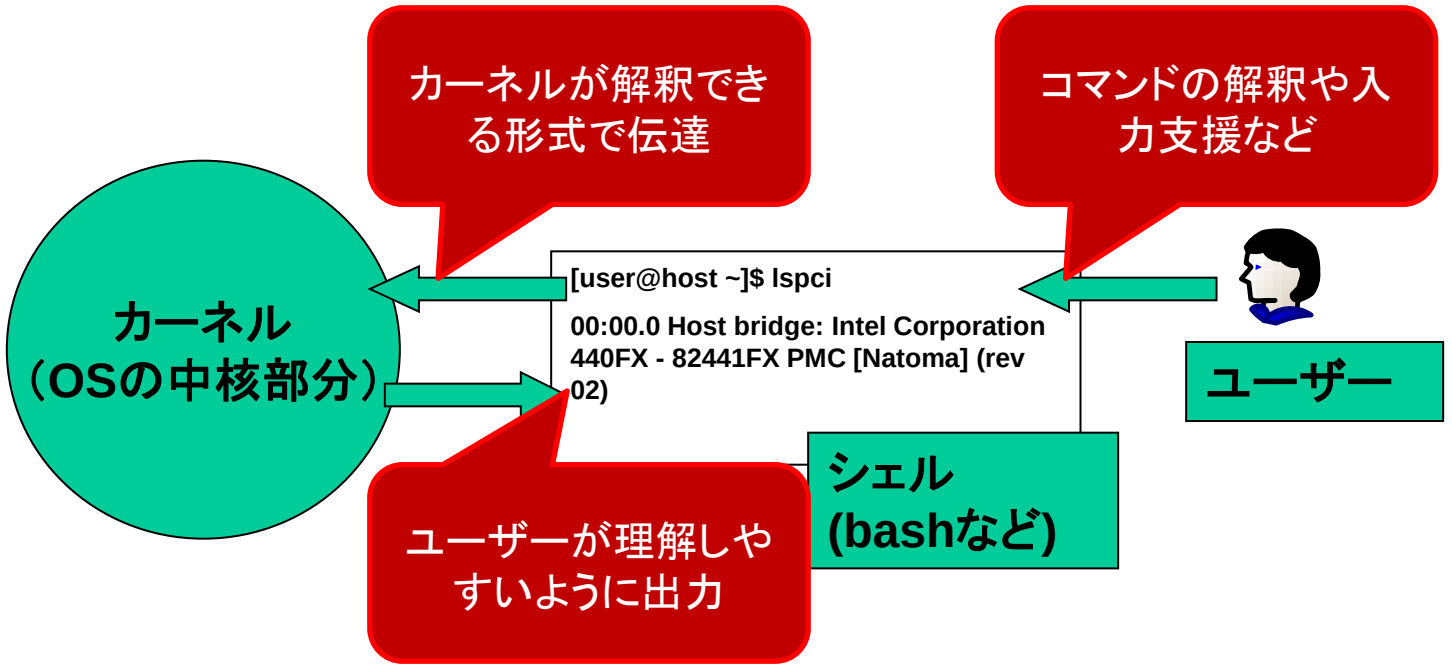
試験範囲概要

- 簡単なBashスクリプトを新規作成できる。
- 作成したBashスクリプトをシステムの利用者に適用できる。
- 処理結果により動作を分岐できるスクリプトを作成できる。
- Linux スキルの無いユーザに規定のコマンドを実行できる環境を提供できる。
- シェルスクリプトの引数を処理できる。

シェルとは

■Linuxではコマンドシェルが用意されており、入力されたコマンドを解釈したり、入力支援機能などを提供します。

- LinuC Lv.1ではbashシェルを前提とした出題範囲がいくつか用意されています。



シェルスクリプトとは

- シェルにはシェルスクリプトと呼ばれるプログラム機能が用意されています。
- コマンドなどをテキストファイルに記述し、記述したコマンドを実行するという処理ができます。それにより、システム管理などの処理を自動化することができます。
- 実行方法
  - bash スクリプトファイル名
  - source スクリプトファイル名
  - . スクリプトファイル名
  - ./スクリプトファイル名

ファイルの読み取り権限が必要

＋
ファイルの実行権限が必要

カレントディレクトリにあるスクリプトであれば、「./～」で指定。他のディレクトリにあるスクリプトであれば、ディレクトリを指定すればよい

スクリプトファイルの実行権限

[root@centos7 ~]# vi test1.sh → test1.shを作成
VAR=test
echo $VAR → この内容を書き込んで保存
[root@centos7 ~]# source test1.sh
test → ファイルの中に書かれたechoコマンドを実行
[root@centos7 ~]# ./test1.sh → この形式では実行権限がないと実行不可
-bash: ./test1.sh: 許可がありません
[root@centos7 ~]# chmod a+x test1.sh
[root@centos7 ~]# ./test1.sh → 実行権限を付与したので、実行できる
test

./～ でスクリプトを実行するには、読み取り＋実行権限が必要
特定のユーザーにだけ実行されたい場合には所有者にだけ付与
すべてのユーザーに実行させたい場合はその他のユーザーに付与

変数について

■変数の定義
変数名＝値

イコールの前後にスペースは入れない

■変数に定義された値を参照
echo $変数名

定義するときには「$」はつけない

■特殊な変数

変数	意味
$1, $2	引数の値
$@, $*	すべての引数
$#	指定した引数の数
$?	直前の処理の成否 ※成功は0
$RANDOM	ランダムな値

▼実行例

```
[root@centos7 ~]# vi argtest.sh
echo $1
echo $@
echo $#
echo $RANDOM
[root@centos7 ~]# chmod a+x argtest.sh
[root@centos7 ~]# ./argtest.sh arg1 arg2
arg1
arg1 arg2
2
16143
```

本来は冒頭に「#!/bin/bash」（シェルの実行のパス）を指定するのが望ましい（シェバン）

0～32767の範囲の乱数を出力（bashのみ）

シェバンが指定されていないときは、ログインシェルにより解釈

制御構文（条件分岐）

■if: 1つの条件で真偽による分岐

```
if [ 条件式 ]
then
    真の場合の処理
else
    偽の場合の処理
fi
```

[コマンド、もしくはtestコマンド
ほか、コマンドを直接指定しても
OK

条件に合致した場合の処理

それ以外の場合の処理
（省略可能）

■条件式で利用できる構文

条件式	意味
値1 -eq 値2	値1と値2が等しい(equal)
値1 -lt 値2	値1が値2より小さい(less than)
値1 -ge 値2	値1が値2以上(greater equal)

■case: 1つの条件で多分岐

```
case 変数名 in
  値1）処理1 ;;
  値2）処理2 ;;
   :
  *）合致しなかった場合の処理;;
esac
```

条件分岐構文の実行例

■if文のサンプル

```
[root@centos7 ~]# vi iftest.sh
#!/bin/bash
if [ $# -eq 2 ]
then
    cp $1 $2
    echo copy complete
fi
[root@centos7 ~]# ./iftest.sh
```

→ 引数が2つ指定されないと処理をしない

```
[root@centos7 ~]# ./iftest.sh iftest.sh
iftest2.sh
copy complete
```

→ 引数を2つ指定すると処理を実行

「#!/bin/bash -x」と指定すると、認識した変数の値などを出力。スクリプトのデバッグなどに便利。

■case文のサンプル

```
[root@centos7 ~]# vi casetest.sh
#!/bin/bash
case $1 in
  y|yes) echo “your input is YES” ;;
  n*) echo “your input is NO” ;;
  *) echo “please input y or n” ;;
esac
[root@centos7 ~]# ./casetest.sh y
your input is YES
```

制御構文（繰り返し）

■for: 値リストによる繰り返し

```
for 変数名 in 値リスト
do
    繰り返す処理
done
```

▼シンプルな構文例

```
[root@centos7 ~]# vi fortest1.sh
#!/bin/bash
for var in aaa bbb ccc
do
        echo $var
done
[root@centos7 ~]# ./fortest1.sh
aaa
bbb
ccc
```

値リスト（スペースで区切る）

■while: 条件文による繰り返し

```
while [条件式]
do
    繰り返す処理
done
```

条件を満たしている間繰り返し

※参考
until: 条件を満たすまで繰り返し

条件分岐構文の実行例

■for文のサンプル

```
[root@centos7 ~]# cat /testbin/fortest.sh
#!/bin/bash
for fname in *.sh
do
        cp $fname ${fname}.bak
        echo copy to ${fname}.bak
done

[root@centos7 ~]# /testbin/fortest.sh
copy to test1.sh
copy to argtest.sh
   :
```

ワイルドカードを利用し、ファイル名をリストに利用できる

${~} で変数の明示

出力されたファイル名のコピーが作成されている

■while文のサンプル

```
[root@centos7 ~]# vi whiletest.sh
#!/bin/bash
i=0
while [ $i -lt 5 ]
do
    echo $i
    i=`expr $i + 1` → $iに1を加える
done
[root@centos7 ~]# ./whiletest.sh
0
1
2
3
4
```

$iの値が5未満の間繰り返し

変数名=`expr 数式`
→ 演算の結果を代入

echoコマンドだと画面に出力するが、他のコマンドを指定して、テストユーザーを追加したり（useradd）、テスト用のファイルを作成（touch）することも可能。

■試験範囲概要

- cronまたはanacronを使用して定期的にジョブを実行したり、atを使用して指定時刻にジョブを実行できる。
  - cronおよびatでジョブを管理する。
    - /var/spool/cron/
  - ユーザがcronおよびatサービスにアクセスできるよう設定する。
    - /etc/cron.allow, /etc/cron.deny
    - /etc/at.deny, /etc/at.allow
    - at, atq, atrm
    - /etc/cron.{d,daily,hourly,monthly,weekly}/
    - /etc/crontab
  - anacronの設定
    - /etc/anacrontab

cronとatの概要

■cron: 定期的に繰り返すジョブを登録

- **crond**により処理
- **crontab**コマンドによりジョブを登録
- **/etc/crontab**に記述されたジョブを実行

■at: 一回限りのジョブを登録

- **atd**により処理
- **at**コマンドによりジョブを登録

crontabコマンド

■-eオプションを指定して実行するとvi（EDITOR変数に登録されているエディタ）が起動し、ファイルを編集

- 保存した内容は「/var/spool/cron/ユーザー名」ファイルに保存

■以下の書式によりジョブを追加

分　時　日　月　曜日　実行内容

- スペースで各列を区切る
- 1行1ジョブ
  - 1つの構文しか実行できないため、スクリプトで記述して追加するのを推奨

■crontabの書式

列	項目	設定範囲
1	分	0～59
2	時	0～23
3	日	1～31
4	月	1～12, jan～dec
5	曜日	0～7（0,7が日曜日）
6	実行するコマンド/スクリプト	

- 「*」で常に実行
- 「*/2」で2分、2時間・・・おき
- 「10,30,50」で複数指定
- 「1-5」で連続値の指定

- 環境によってPATHは/binと/usr/binしか通っていないことがあるので注意

cronジョブの追加例

- ＊ ＊ ＊ ＊ ＊ touch `date +%H%M.txt`
  - 毎分、「1345.txt」（時分.txt）という空のファイルを作成
- 0 ＊/2 ＊ ＊ ＊ /usr/sbin/ntpdate pool.ntp.org
  - 2時間おきにNTPサーバーと同期
  - 環境によっては、/usr/sbin/ などに格納されている実行ファイルはフルパスで記述しないと認識しない（スクリプトの中で、PATHを追加などの対応でもOK）
  - 「＊ ＊/2 ＊ ＊ ＊」だと2時間おきに0～59分の60回ジョブが実行される
    →0:00～0:59の60回の後、2:00～2:59の60回・・

ジョブに利用しているdateコマンドやntpdateコマンドは、「1.09：重要なシステムサービス」で出題される内容です。

cronジョブの追加例

0 23 * * 1-5 /tmp/backup.sh

- 平日の23時ちょうどに/tmp/backup.shというスクリプトを実行

```
[root@centos7 ~]# cat /tmp/backup.sh
#!/bin/bash
tar cvf /tmp/home-backup-`date +%m%d`.tar /home
```

- /homeを/tmp/home-backup-1031（月日）.tar にバックアップ

■次のようなジョブの実行を想定

- システムが24時間起動していなくても、1日1回ジョブを実行
  - cronジョブは、その時間にシステムが起動していないとスキップされる
- 1日のどこかのタイミングでジョブが起動されればいい
  - 厳密な時間の設定はできない

anacron用のデーモンは常駐せず、crondにより実行

■/etc/anacrontabにより設定（以下は抜粋した内容）

- 以前はcrontabで管理されており、指定時刻に起動していないとスキップ

```
#period in days   delay in minutes   job-identifier   command
1       5         cron.daily           nice run-parts /etc/cron.daily
7       25        cron.weekly          nice run-parts /etc/cron.weekly
@monthly 45       cron.monthly         nice run-parts /etc/cron.monthly
[root@centos7 ~]# ls /etc/cron.daily/
logrotate  man-db.cron  mlocate
```

1日1回、/etc/cron.daily/に格納されたスクリプトを実行
- run-parts: 指定したディレクトリのスクリプトを実行

ログのローテーション処理

実行した日は/var/spool/anacron/ 以下のファイルに記録

atコマンド

■書式：at [オプション] 日時

- -fでファイルを指定しなければ、専用のプロンプトが表示され、実行する内容を直接入力。

オプション	説明
-l (atq)	ジョブの一覧表示
-c	指定したジョブの内容を表示
-d (atrm)	ジョブの削除
-f	ファイルを指定

▼実行例

```
[root@centos7 ~]# at 14:00
at> touch attest.txt
at> <EOT> → ctrl + dを押して、登録完了
job 1 at Mon Oct 26 14:00:00 2020
[root@centos7 ~]# at -l
1       Mon Oct 26 14:00:00 2020 a root
```

【参考】cron, atのアクセス制御

1. /etc/～.allowファイルがある
→ 記述されたユーザーが実行可能

2. /etc/～.denyファイルがある
→ 記述された以外のユーザーが実行可能

3. どちらもない
→ rootユーザーのみ（例外あり）

cronの場合、IEEE Std 1003.1-2001準拠でなければ、すべてのユーザーが実行可能。ただし、そのような場合でも空の/etc/cron.denyファイルが用意されていれば、すべてのユーザーが実行可能

■ご質問がありましたら、よろしくお願いいたします。

- 試験を受ける前に、試験範囲の詳細をご覧になっていない場合、一度ご覧になることをおすすめします。
  https://linuc.org/linuc1/range/101.html
  https://linuc.org/linuc1/range/102.html
  - 各項目に「重要度」あり
  - 出題範囲に明記されているコマンドや設定ファイルは必ず把握
  - 受験するバージョンに即した教科書の利用をおすすめします

おわりに

■ご参加いただき、ありがとうございました。

■答えきれなかったご質問については、後日YouTube動画公開の際に掲載させていただきます。

「最短突破 LinuCレベル1 合格教本 ver.10対応」(技術評論社)
好評発売中です。
※ver4対応の旧版も引き続き発売中です