段位制IT資格道場

LinuC レベル2 201試験 学習用テキスト

予備知識ゼロから合格ラインへ —— 読んで理解する1冊目

無料サンプル(第1章まで)

LPI-Japan公式 LinuC レベル2 201試験 出題範囲 Version 10.0 準拠
2026年7月版

本書のご利用にあたって

- 本書は「段位制IT資格道場」(www.shikaku-doujou.com)の購入者限定テキストのうち、学習用テキストの第1章までを収録した**無料サンプル**です。どなたでもご覧いただけます。
- 無料サンプルであっても著作物です。本書の転載・改変しての再配布・二次販売を禁じます。
- 本書を AI モデルの学習データとして利用すること(学習目的の入力・データセット化を含む)を禁じます。
- 製品版(学習用テキスト・暗記用ノート・頻出度別ノートの3点)は、ご購入者を識別できる透かし入りのPDFとして提供されます。
- 本PDFはスマートフォン・タブレットでの閲覧に合わせたA5版面・文字サイズで組版しています。

不正な配布を見つけた場合は info@shikaku-doujou.com までご連絡ください。

目次

はじめに —— この教材の使い方

このテキストは、LPI-Japan が実施する **LinuC レベル2 201試験** に合格することを目的に作られています。

レベル1 との違いは視点の高さです。レベル1 は「動いている Linux サーバーを操作・運用する」知識を問う試験でした。201試験が問うのはその一段手前 —— **Linux サーバーを設計し、組み立てる側**の知識です。電源を入れてからログインプロンプトが出るまでに何が起きているのか、ディスクをどう切ってどう見せるのか、壊れたときにどう切り分けるのか、仮想マシンとコンテナをどう作って動かすのか。「使えます」ではなく「組めます」と言えるかどうかが問われます。

覚え方も変わります。レベル1 では「このコマンドは何をするか」で足りましたが、201試験では次の 3 点まで踏み込む必要があります。

1. **似ているコマンドの役割の違い**(例: `grub2-install` と `grub2-mkconfig`、`lvextend` と `resize2fs`)
2. **手順の順番**(例: `pvcreate` → `vgcreate` → `lvcreate` を飛ばせない理由)
3. **設定が「今だけ」効くのか「次回起動後も」効くのか**(例: `sysctl -w` と `/etc/sysctl.conf`)

本書はこの 3 点に紙面を集中させます。網羅的な man ページの翻訳は目指しません。

試験の基本情報

項目	内容
試験名	LinuC レベル2 201試験

項目	内容
運営	LPI-Japan
前提条件	LinuC レベル1 の認定保有(101試験・102試験の合格による認定)
認定要件	201試験と202試験の両方に合格するとLinuC レベル2 に認定される(両試験は 5 年以内に合格する必要がある)
問題数	約 60 問
試験時間	90 分(うち試験後アンケートを 5 分含むため、解答時間は実質 85 分)
合格ライン	**非公開**(公式にスコア基準は公表されていない)
受験料	16,500 円(税込)※1 試験あたり。2026 年 7 月時点
試験言語	日本語・英語
受験方法	ピアソン VUE 経由の CBT(テストセンターまたは自宅)。企業向けに PBT による団体受験制度もある
認定の有効期間	認定日から 5 年間「ACTIVE」として記録される

補足を 3 つ。**合格ラインは公表されていません**。「何点で受かる」という数字は公式に存在しないので、本書でも一切示しません。**出題比重もパーセンテージでは公表されていません** —— 公式は各トピックに「重要度」という数値を割り振る形で比重を示しています。そして**認定の維持**。有意性を保つには、5 年以内に同一レベルの最新バージョン試験に合格するか、上位レベルの新規

認定を取得する必要があります。なお受験料や試験仕様は改定されることがあるので、申し込み前に公式サイトで最新情報を確認してください。

重要 —— RAID は 201試験の出題範囲に含まれない

先に、時間の使い方に直結する話をします。**201 試験の出題範囲に RAID(`mdadm` によるソフトウェア RAID 構成)は含まれません。**

Linux の資格というと「ストレージといえば RAID と LVM」という思い込みが働きがちで、他系統の資格や古い学習ノートには `mdadm` が必ず出てきます。しかし本試験のストレージ管理の主題区分は「ファイルシステムの設定とマウント」「ファイルシステムの管理」「LVM によるボリューム管理」の 3 つで、RAID の構成は入っていません。本サイトの問題集にも `mdadm` を扱う設問は 1 問も収録していません。

したがって本書でも RAID の解説はしません。**浮いた時間は LVM に回してください。**LVM は出題範囲に明記されており、コマンドの順番と拡張手順という「手が動くかどうか」を問いやすい題材なので、投資対効果が最も高い領域です。

本書の構成(全 9 章)

公式の出題範囲(Version 10.0)の 6 テーマにそのまま沿い、内容の多いテーマを 2 章に割っています。

章	扱う内容	対応テーマ
第1章・第2章	起動・GRUB・systemd / カーネル	① システムの起動と Linux カーネル
第3章・第4章	マウントとファイルシステム / LVM	② ファイルシステムとストレージ管理

章	扱う内容	対応テーマ
第5章	ネットワーク構成と切り分け	③ ネットワーク構成
第6章・第7章	ビルドとバックアップ / 監視と自動化	④ システムの保守と運用管理
第8章	仮想化サーバーと KVM	⑤ 仮想化サーバー
第9章	コンテナ	⑥ コンテナ

3 ステップ学習法

- **STEP 1(理解する)**: 本書「学習用テキスト」を通読する。ここで全体の地図を頭に入れる
- **STEP 2(覚える)**: 「暗記用ノート」の一問一答を、即答できるまで繰り返す
- **STEP 3(仕上げる)**: 「頻出度別ノート」で頻出度 A の論点から総点検し、仕上げに本サイトの問題演習を回す

本試験は選択式に加えて、**コマンドを実際にキーボードから入力させる形式**が出ます。本サイトの問題集もこの入力形式に対応しています。読んで分かった気になっただけでは手が動かないので、STEP 3 では必ず入力形式も回してください。

第1章 システムの起動 —— ブートプロセスとGRUB、そしてsystemd (出題テーマ①その1)

電源ボタンを押してからログインプロンプトが出るまでの一部始終を追いかけます。**201試験で最も問われる領域のひとつ**で、「壊れたときにどこを見るか」がそのまま設問になります。

全体を貫く考え方はひとつ —— **起動とは、小さな担当者が次の担当者にバトンを渡していく連鎖**です。誰が誰に渡すのかを覚えれば、「どこで止まったか」から「何を直すか」を逆算できます。

1-1 電源投入からログインまでの全体像

BIOS ベースの x86 システムでは、典型的にはこう進みます。

1. **電源投入 → ファームウェア(BIOS)が起動し、POST(自己診断)を実行**
2. **ディスク先頭の MBR にあるブートストラップコードを実行**
3. **ブートローダー(GRUB)がカーネルと initramfs をメモリへ読み込む**
4. **カーネルが起動し、initramfs を使って本来のルートファイルシステムをマウントし、そちらへ切り替える**
5. **最初のプロセス(systemd)が起動し、各種サービスを立ち上げる**

順番を問う設問が実際に出ます。「initramfs はカーネルより前か後か」で迷いやすいのですが、**initramfs はカーネルが読み込んで使うもの**なので「カーネルと一緒にメモリへ載る」と覚えてください。

1-2 BIOS/MBR 方式と UEFI/GPT 方式

MBR(マスターブートレコード)

BIOS 方式では、ディスク先頭セクタ(LBA0)の **MBR** から起動します。MBR は **512 バイト**ちょうどで、内訳が問われます。

部分	サイズ	役割
ブートストラップコード	446 バイト	最初に実行される起動用のコード
パーティションテーブル	64 バイト	基本パーティション 4 つ分の情報
ブートシグネチャ	2 バイト	`0x55AA`。有効な MBR である印

446 + 64 + 2 = 512。この足し算ごと覚えるのが確実です。

ESP(EFI System Partition)

UEFI 方式では、ファームウェアが**パーティション上に置かれた実行ファイルを直接起動**します。その置き場が **ESP** です。

- ファイルシステムは **FAT32(vfat)** である必要がある
- 多くのディストリビューションでは `/boot/efi` にマウントされる
- 配下の `/EFI/<ディストリビューション名>/` に `grubx64.efi` などが置かれる

`/boot/efi/` と `/boot/grub2/`(Debian 系では `/boot/grub/`)の違いは定番の設問です。前者は ESP そのもので `.efi` ファイルが入る場所、後者は GRUB2 の設定ファイル `grub.cfg` やモジュールが入る場所 —— **別物**です。

UEFI のブートエントリと efibootmgr

「どのローダーをどの順番で試すか」はファームウェアの **NVRAM** に記録されます。これを操作するのが `efibootmgr` です。

コマンド	動作
`efibootmgr -v`	各エントリが指すローダーのパスまで含めて詳細表示
`efibootmgr -o 0003,0000,0002`	恒久的なブート順序(BootOrder)を指定した順に設定
`efibootmgr -n 0003`	次回 1 回限りのブートエントリを指定(BootOrder は変えない)

`-o` は恒久、`-n` は次回のみ —— 取り違えが狙われます。

GPT + BIOS レガシーブート、そして Secure Boot

GPT のディスクを UEFI ではなく BIOS レガシーブートで起動させる場合、GRUB のコアイメージを置く余地が単純には無いため、**BIOS Boot Partition** という専用パーティションを用意します。パーティションタイプに `bios_grub` **フラグ**を立て、**ファイルシステムは持たず**、サイズは数百 KB 〜 1 MB 程度です。

UEFI Secure Boot が有効だと、署名されていないブートローダーは起動できません。多くのディストリビューションは **shim** という小さなブートローダーを間に挟んで回避しています。shim は Microsoft によって署名されているのでファームウェアから起動でき、shim 自身の鍵または **MOK(Machine Owner Key)** として登録した鍵で GRUB 本体を検証してから起動します。

1-3 GRUB 2 —— 設定ファイルは「書くもの」ではなく「生成物」

最重要の考え方をひとつ。

> **`grub.cfg` は手で編集しない。`/etc/default/grub` と `/etc/grub.d/` を編集し、生成コマンドで作り直す。**

理由は、`grub.cfg` が自動生成されるファイルだからです。カーネルを更新するとパッケージ側が生成コマンドを走らせるため、**手編集した内容はそのとき消えます**。この「なぜ手編集してはいけないか」がそのまま設問になります。

ファイル/コマンド	役割
`/etc/default/grub`	タイムアウトやカーネル起動パラメータなど、値としての設定
`/etc/grub.d/`	メニュー項目を組み立てるスクリプト群
生成コマンド	上記 2 つを読み取り、`grub.cfg` を組み立てる
`grub.cfg`	GRUB 2 が実際に読む最終的な設定ファイル(生成物)

生成コマンドはディストリビューションで名前が違う

系統	生成コマンド	出力先の例
RHEL/CentOS 系	`grub2-mkconfig -o /boot/grub2/grub.cfg`	`/boot/grub2/grub.cfg`
Debian/Ubuntu 系	`grub-mkconfig -o /boot/grub/grub.cfg`	`/boot/grub/grub.cfg`

系統	生成コマンド	出力先の例
Debian/Ubuntu 系(ラッパー)	`update-grub`	同上

違いは「2」が付くか付かないかだけで機能は同じです。`update-grub` は Debian/Ubuntu 系のラッパースクリプトで **RHEL/CentOS 系には存在せず**、逆に `grub2-mkconfig` は Debian/Ubuntu 系には存在しません。`update-grub2` というコマンドは**存在しません**。

最頻出のひっかけ：`-o`（`--output`）を付けずに `grub2-mkconfig /boot/grub2/grub.cfg` と打っても、生成結果は**標準出力に表示されるだけ**でファイルは更新されません。

grub-install との役割の違い

`grub-install`（RHEL 系では `grub2-install`）は、**ブートローダー本体をディスクに書き込む**コマンドです。BIOS/MBR 方式なら MBR とその周辺へ、UEFI 方式なら ESP 上へ GRUB のイメージやモジュールを配置します。**`grub.cfg` の生成は一切行いません。**

コマンド	何をするか
`grub2-install /dev/sda`	ブートローダー**本体**をデバイスへインストールする
`grub2-mkconfig -o ...`	**設定ファイル** `grub.cfg` を生成する

「本体を置く」のか「設定を作る」のか、と自問してから選んでください。

同じマシンに Windows や別のディストリビューションが入っている場合、それらをメニューに自動追加してくれるのが **os-prober** です。生成コマンド実行

時に他パーティション上の OS を検出し、`/etc/grub.d/` 内のスクリプトを通じて `grub.cfg` に反映されます。

1-4 起動できなくなったときの復旧

grub rescue> プロンプト

`grub.cfg` が消えるなどして GRUB が通常メニューを出せないと `grub rescue>` で停止します。復旧手順が問われます。

1. `ls` でデバイスとパーティションの一覧を確認する
2. `set root=(hdX,Y)` でルートパーティションを指定する
3. `insmod normal` で通常モード用のモジュールを読み込む
4. `normal` を実行して通常モードへ復帰する

「`ls` で探し、`set root` で教え、`insmod normal` → `normal` で戻す」という流れです。

UEFI Shell

UEFI ファームウェアが提供する簡易的なコマンドライン環境です。起動可能な OS やブートローダーが見つからないときに直接起動でき、ファイルシステムを参照して手動でブートローダーを実行したり、ブート関連の変数を確認したりできます。**NVRAM のブートエントリが失われた**ときの最後の手段として登場します。

1-5 initramfs —— ルートをマウントするための足場

initramfs(古い形式は initrd)は**一時的な小さなルートファイルシステム**です。カーネルと一緒にメモリへ読み込まれ、本来のルートファイルシステムをマ

ウントするために必要なモジュールやツールを提供し、役目を終えると `switch_root` などで本来のルートへ制御を移して退場します。

なぜ必要か。ルートファイルシステムが次のような場所にあると、カーネル単体ではマウントできないからです。

- **LVM の論理ボリューム**上にある
- **LUKS で暗号化**されている
- カーネルに組み込まれていない**特殊なストレージコントローラ**の先にある

鍵のかかった部屋に入るために、まず玄関へ道具箱を置いておく —— その道具箱がストレージ用モジュールとツール類にあたります。生成ツールは複数あり、`dracut` が多くの主要ディストリビューションで採用されている現代的な枠組み、`mkinitrd` は主に旧来の Red Hat 系、`mkinitramfs` は主に Debian 系で使われてきました。

1-6 systemd —— ユニットとターゲット

ユニットファイルと優先順位

systemd の管理対象は **ユニット**で、`.service`・`.target`・`.mount`・`.timer` などの種類があります。ユニットファイルは複数のディレクトリに置かれ、**同名なら優先順位が決まっています**。

場所	位置づけ	優先度
`/etc/systemd/system/`	管理者が置く設定	最も高い
`/run/systemd/system/`	実行時に動的生成される設定	中

場所	位置づけ	優先度
`/usr/lib/systemd/system/`	パッケージが提供する既定の設定	最も低い

`/etc/` にあるものがパッケージ提供のものを上書きするという一点で正解できます。「複数あると自動でマージされる」は誤りです。

ターゲットと旧ランレベル

ターゲット	相当するランレベル	内容
`multi-user.target`	3	CUI での通常のマルチユーザー環境
`graphical.target`	5	GUI 付きのマルチユーザー環境
`rescue.target`	1 相当	ローカルファイルシステムをマウントした上で単一のシェルを起動
`emergency.target`	——	ファイルシステムのマウント処理をほぼ行わない最小構成のシェル

`rescue` と `emergency` の違いが最重要です。`rescue.target` は `local-fs.target` などによって**ローカルファイルシステムをマウントしてから**シェルを起動し、`emergency.target` はそれすら行いません。「より深刻なほうが emergency」と覚えます。

依存関係 —— Wants= と Requires=

- `Requires=`：強い依存。依存ユニットの起動に失敗すると**自分自身も起動に失敗しうる**
- `Wants=`：弱い依存。依存ユニットが失敗しても**自分の起動には影響しない**

「あると嬉しい(Wants)」「無いと困る(Requires)」という語感がそのまま強弱に対応します。

1-7 systemctl —— 必修コマンド

起動・自動起動・禁止の 3 段階

201試験で最もよく問われる部分です。**「今」の話と「次回起動後」の話、そして「禁止」を切り分けてください。**

コマンド	効く時点	内容
`systemctl start <unit>`	今すぐ	即座に起動する。次回起動時の設定は変えない
`systemctl enable <unit>`	次回起動から	自動起動用のシンボリックリンクを作る。今は起動しない
`systemctl disable <unit>`	次回起動から	自動起動リンクを消す。**手動 start は依然できる**
`systemctl mask <unit>`	今も次回も	`/dev/null` へのリンクに置き換え、**手動 start も含めて起動不能にする**

`disable` と `mask` **の違い**は超頻出です。「root が手で叩いても、他のユニットからの依存経由でも起動させたくない」なら `mask`。解除は `systemctl unmask` です。

状態を見る・反映する・切り替える

コマンド	見えるもの/動作
`systemctl status <unit>`	現在の状態・ユニットファイルの場所・直近のログ
`systemctl list-units`	**メモリ上にロード済み**のユニットと状態(active/inactive)
`systemctl list-unit-files`	**ディスク上に存在する全ユニットファイル**とenabled/disabled
`systemctl is-enabled <unit>`	自動起動が有効かだけを簡潔に返す(スクリプト向き)
`systemctl cat <unit>`	本体のユニットファイルとドロップイン設定をまとめて表示
`systemctl daemon-reload`	編集したユニットファイルを systemd に読み直させる
`systemctl set-default graphical.target`	**次回起動から**の既定ターゲットを変更
`systemctl isolate rescue.target`	**今すぐ**現在のセッションを指定ターゲットへ切り替え

`list-units` は「今ロードされているもの」、`list-unit-files` は「ファイルとして存在するもの全部」。絞り込みは `systemctl list-units --type=service --state=active` のように書きます。

`daemon-reload` を忘れると、編集したのに古い定義のまま動きます。「変更が反映されない」という設問の答えはたいていこれです。`set-default` は恒久、`isolate` は即時 —— ここも「今か次回か」の切り分けです。

`systemctl status` の出力の読み方も問われます。`Loaded:` 行の `(/usr/lib/systemd/system/sshd.service; enabled; ...)` から「ユニットファイルの場所」と「自動起動が有効」を、`Active: active (running)` から「現在稼働中」を読み取れれば正解できます。

起動が遅いときは **`systemd-analyze blame`** で、初期化に時間のかかっているユニットを所要時間順に一覧できます。

第1章の試験ポイント

- MBR は 512 バイト = ブートストラップ 446 + パーティションテーブル 64 + シグネチャ 2(`0x55AA`)
- ESP は FAT32・多くの環境で `/boot/efi` 。`efibootmgr` の `-o` は恒久、`-n` は次回のみ
- `grub2-install` = ブートローダー**本体**の設置、`grub2-mkconfig` = **設定ファイル**の生成。`-o` 無しは標準出力に出るだけ
- `grub.cfg` は手編集しない(再生成で消える)。`update-grub` は Debian 系だけのラッパー
- initramfs は本来のルート(LVM・暗号化・特殊コントローラ上など)をマウントするための一時環境
- `disable` は自動起動を止めるだけ、`mask` は手動 start も含めて起動不能。解除は `unmask`

- `rescue.target` はファイルシステムをマウントする、`emergency.target` はほぼ行わない
- ユニットファイルは `/etc/systemd/system/` が最優先。編集後は `daemon-reload`

サンプルはここまでです

続き(第2章〜巻末)は、LinuC 201 問題集のご購入で、暗記用ノート・頻出度別ノートと合わせて3点すべてダウンロードできます。

LinuC レベル2 201試験 学習用テキスト(無料サンプル)

版
2026年7月版(LPI-Japan公式 LinuC レベル2 201試験 出題範囲 Version 10.0 準拠)

発行
段位制IT資格道場(https://www.shikaku-doujou.com)

お問い合わせ
info@shikaku-doujou.com