Arch Linux インストールガイド

首都大学東京マイクロコンピュータ研究会
nosada

目次

第 1 章

はじめに

1.1 概要

「Arch Linux インストールガイド」（以下「この文書」と表記）は、首都大学東京マイクロコンピュータ研究会（以下マイコン研）の一企画であるマイコン研 Linux ゼミ（以下「本ゼミ」と表記）で使用するために作成されたものです。

本ゼミの目標は以下のとおりです。

- Linux とは何たるかを知り、GNU/Linux についての知識を身に着ける
- 実際に Arch Linux を導入して GNU/Linux[*1] を使用してみる
- 導入した Arch Linux をデスクトップ用 OS としてセットアップし、デスクトップ用 OS として通常用途に使用できるようにする
- 自由かつ無料な OS を通じてコンピュータを眺め、新たな価値観を手にする。

本ゼミではこれらの目標を達成するために、まず GNU/Linux に至るまでの OS の歴史を概説し（第 2章）、続いて Arch Linux をインストール、セットアップし（第 3章）、デスクトップ OS として実用に耐えうるであろうと思われる程度までパッケージの追加、設定を行う（第 4章）ことを目的としています。この文書は以上の目的を達成するために、その内容及び方法、要する知識などを示すものです。

この文書内における各章の役割は以下のとおりです。

- 第 1章：はじめに
  - 本ゼミの内容及びこの文書の概要と動作環境を示します。
- 第 2章：Linux とは？
  - GNU/Linux を使用する前に、Linux および GNU/Linux についての説明を与えます。
    なおこの章は本ゼミの主目標である Arch Linux のインストールには直接は関係しませんので、読み飛ばしても結構です。
- 第 3章：Arch Linux のインストール
  - Linux ディストリビューションのひとつである Arch Linux についての説明を与え、そのインストールイメージの入手、インストール、インストール後に必要な設定などについての説明を与えます。
    インストールに必要となる Unix コマンドやツール等の説明もこの章で行います。
- 第 4章：Arch Linux を使用する
  - Arch Linux をデスクトップ用途に使用するための各種ソフトウェアのインストールや設定などを行います。

[*1] Linux 自体は OS ではない。一般に “Linux” と呼ばれる OS の実体は GNU/Linux と呼ばれるシステムである。この話題については第 2章で扱う。

なお、仮想マシンを利用して GNU/Linux をインストールする際には、別途「VirtualBox を用いた仮想マシンの構築」[*2] をご参照下さい。

1.2　環境

前述の通り、本ゼミでは Linux ディストリビューションとして Arch Linux[*3] を使用します。

本ゼミの参加者の推奨環境は表 1.2aの通りとなります。現在通常の手段で手に入る PC であれば、おそらくこの環境を満たしているはずです。仮想マシンを利用せずに実機へのインストールを行う場合ならば、推奨環境よりも低い性能のマシンでも動作は可能でしょう。もちろんその場合、CPU も 64 ビットアーキテクチャであることを要求されませんが、この文書では GNU/Linux を導入するマシンの CPU は 64 ビットアーキテクチャであることを想定しています。また、仮想化環境を使用しない場合、勿論インストール対象の PC に搭載されている CPU が仮想化に対応している必要はありません。

参考までに、筆者の環境は表 1.2bの通りです。この環境下での動作を確認しましたし、現在もしています。

表 1.1: 動作環境

OS	仮想化環境を使用する場合、それが動作するもの
CPU	仮想化対応製品かつ 64 ビットアーキテクチャ (x86_64、x64、amd64) のもの
RAM	可能な限りたくさん (2GB 以上あれば十二分)
ディスク	可能な限りたくさん（空き容量が 10GB 以上あれば十二分）

(a) 推奨環境

OS	Arch Linux
CPU	Intel® Core™ 2 Duo P8700 2.53GHz
RAM	8GB
ディスク	60GB

(b) 筆者の環境

1.3　文書生成環境

この文章は以下の環境で作成されました。以下に示すソフトウェアは、全て無料で入手、使用できます。詳細は各ソフトウェアに関するライセンスをご参照下さい。

- ワードプロセッサソフト
  LyX[*4] version 2.1.0
- 組版ソフトウェア
  TeXLive 2013[*5] version 3.1415926-p3.4-110825-2.6(pTeX)
- フォント

[*2] この文書が入っているディレクトリの中にある VBoxGuide.pdf が該当する。
[*3] https://www.archlinux.org/
[*4] http://www.lyx.org/
[*5] http://www.tug.org/texlive/

梅フォント[*6] 梅明朝 S3(ume-tms3.ttf) 及び梅ゴシック S5(ume-tgs5.ttf)

[*6] http://sourceforge.jp/projects/ume-font/

第 2 章

Linux とは？

第 1章において、なんの説明も加えることなく、「Linux ディストリビューション」という言葉を利用しました。Arch Linux とは数ある Linux ディストリビューションの一つです。そして Linux ディストリビューションは OS を指すものではありません。ではこの「Linux ディストリビューション」の意味するものは何でしょうか。

また、Linux ディストリビューションは数あれど、“Linux” というソフトウェアは “Linux” のみです。では Linux とは一体何でしょうか。Linux は一般に OS であるかのように扱われていますが、実は OS ではありません。一般に OS として Linux と呼称するとき、その実体は GNU/Linux と呼ばれるシステムを指します。Linux とは OS を構成する部品の一つであるカーネルと呼ばれるものです。ではカーネルとは何でしょうか。そもそも OS とは何でしょうか。

本章では、これらの疑問に対し説明を与えることを目的とします。

2.1 OS

OS(Operating System) は、ハードウェアを管理し操作するためのソフトウェアです。OS はハードウェア、即ちコンピュータの五大装置と称される演算装置、制御装置、記憶装置、入力装置、出力装置すべてを効率よく管理し、またその機能を便利に扱えるようにすることが期待されます。これらを達成するために、OS はハードウェアの管理だけでなく、ソフトウェア[*1]API(application programming interface)[*2] や ABI(application binary interface)[*3] の整備、また OS 上で動作するプロセス[*4] の管理全般等、コンピュータをハードウェアの方面からもソフトウェアの方面からも管轄する大規模なソフトウェアとなっています。ざっくばらんにいえば、ソフトウェアを動作させるための基盤、縁の下の力持ちということになるでしょう[*5]。

2.2 カーネル

2.1で、OS はハードウェアを管理し操作するソフトウェアだと説明しましたが、これは詳しく見ると実は誤りです。確かに OS はハードウェアを管理し操作するのですが、実際には OS 内部に存在する**カーネル (kernel)** がその役割を担っています。カーネルは OS の中核たる存在であり、通常は OS を構成する部品の一つです。カーネルは OS によっ

[*1] ソフトウェアという概念は動作する場所（システム側かユーザ側か）や用途によって分類されることがあるが、ここでは分類せず混同して扱う。

[*2] ソフトウェアから呼び出し可能な機能を定義し、提供するもの [1]。ソフトウェアにおいてソースコードの側から利用される関数等がそれで、C でいう `printf` や `scanf` 等の定義が API にあたる。

[*3] ソフトウェアがソフトウェア自身や OS、ライブラリと何らかの情報をやりとりする際に使用する方法を定義するもの [1]。API の例を用いれば、`printf` や `scanf` が実行された際にどのような動作をするか、またさせられるかという事柄の定義が ABI となる。

[*4] 一般に実行中のプログラムを指す。

[*5] OS の厳密な説明は大変厄介で、このようなモヤっとした説明でお茶を濁すしか無い（筆者の浅学が原因でもある）。厳密な知識を求める場合は [2] などを参照することが望ましい。

てシステム起動時に真っ先に（つまり一番目に）起動されるプログラムであり、2.1で取り上げた OS の役割のうちのほとんどを担当します。つまりその具体的な役目は、ハードウェア全般の管理、ソフトウェアのお世話、コンピュータ資源（メモリ、ストレージ、あるいは実行中のプロセス等）の効率的な運用が挙げられます。

2.3　Linux カーネル

Linux カーネル (Linux kernel) は 1991 年、当時ヘルシンキ大学（フィンランド）の学生であったリーナス・トーバルズ（Linus Benedict Torvalds）によって書かれたプログラムから誕生しました。

Linux の歴史を紐解くには、Unix という OS と GNU プロジェクトについての知識が必要になります。よって、この節では Unix と GNU の歴史も併せて振り返ることにします。

2.3.1　Unix

Unix（商標上は “UNIX” が正しい [8]）とは、1969 年に、当時 AT&T[*6] に属していたベル研究所[*7] の研究員であったケン・トンプソン (Kenneth Lane Thompson)、デニス・リッチー (Dennis MacAlistair Ritchie) らによって開発された OS です。Unix は最初アセンブリ言語で記述されており[*8]、お世辞にも扱いやすいとは言えないものでしたが、後に高級言語[*9] である C 言語で記述し直され、より扱いやすく、また Unix 開発者が想定したハードウェア以外のハードウェアへ移植することも容易なプログラムになりました。

Unix は AT&T 傘下のベル研究所で開発されたものであり、通常では Unix についての権利はすべてベル研究所、ひいては AT&T が持つものですが、AT&T は当時法的制限によりコンピュータ関連の事業への参入が禁じられており、加えて保有する電話関連以外の技術は独占を禁じられていました [7]。これにより、AT&T は希望する者全てに Unix のソースコードを無償で提供することとなり、自由なソフトウェアとなりました。結果、開発者の手を離れて多数の大学や企業に渡り、多くの改善や改良を加えられることで、次第に成長していくこととなります。しかし、電話事業を独占していた AT&T は 1984 年に事業の分割を余儀なくされ、これによりベル研究所は AT&T の手を離れ別の企業の傘下に加わります。この事業の再分割の結果、AT&T はコンピュータ関連の事業への参入が再び認められるようになりました。Unix を自由の身にしていたのは AT&T に課せられた法的制限によるものです。いまや AT&T にはその法的制限はなく、無償で配布する道理はありません。なぜなら AT&T は利益を求める企業であり、Unix は既に市場を形成している、魅力的な商品であるからです。この結果、AT&T は Unix の無償配布をやめ、商用 Unix として、Unix を自社の商品としたのでした。

今まで無償で提供されてきた Unix が有償となる、これは Unix の利用者にしてみれば大問題です。今までタダで使えたものがいきなり有料になったら誰だってびっくりしますよね。一回ググるたびに Google から 5 円請求されるようになればおそらく文明社会に生きている人達はびっくり仰天するでしょう。規模は違えど、Unix 有料化は Unix に親しんだ人達を仰天させたことでしょう。結果、Unix の利用者は、AT&T が販売する商用 Unix と互換性を持つ OS を開発することになります。これが **Unix 互換**の OS といわれるものになります。Linux も Unix 互換の OS であると言われますが、その成り立ちは Unix とは異なります。

[*6] American Telephone and Telegraph Company、電話を発明したグラハム・ベル (Alexander Graham Bell) が電話の特許を保持するために 1874 年に設立した権利団体を前身とする企業（`http://en.wikipedia.org/wiki/AT&T_Corporation`より）。

[*7] アメリカにある研究機関。トランジスタも C 言語もこの研究所から誕生した。

[*8] 当時はシステム開発に使えるような高級言語が存在せず、ハードウェアに密接に関わる泥臭い言語でプログラムを作成する必要があった。現在もハードウェアに密接に関わる部分ではアセンブリ言語が生きているが、その量は当時よりも明らかに少ない（はず）。

[*9] プログラミング言語のうち、より自然言語（人間が用いる言語体系）に近いものを高級言語といい、より機械語（コンピュータが用いる記号列の体系）に近いものを低級言語という。

2.3.2　BSD

BSD(Berkeley software distribution) とは、アメリカ、カリフォルニア大学バークレー校 (University of California, Berkeley : UCB) の computer system research group(CSRG) が、1977 年から 1995 年まで開発、配布していたソフトウェア群と OS を指します [14]。

BSD[*10] は当時はまだ無償で配布されていたベル研究所の Unix を元に開発された OS で、BSD Unix と呼ばれた正真正銘の Unix でした。1984 年の AT&T の事業再編までは BSD は自由に公開されており、ベル研究所の Unix と同様多数の大学や企業で使用されていましたが、ベル研究所の AT&T からの分離以降、BSD はライセンスの問題で自由に公開することが不可能になりました。このため、CSRG は AT&T のライセンスが関わってくる AT&T 由来のコードを BSD から削り、必要な部分を書き足し、AT&T のライセンスに抵触しない Unix 互換の OS として BSD を開発することにしました (4.3BSD Net/2)。この 4.3BSD/Net2 を元に、1985 年に出現した 32 ビット CPU である Intel 80386 での動作を目的とした 386BSD がウィリアム・ジョリッツ (William Frederick (Bill) Jolitz) とリン・ジョリッツ (Lynne Greer Jolitz) により開発されましたが、この公開は 1992 年でした。というのも、4.3BSD はこの時点では OS としては不十分であり、完全に動作するものではなく、386BSD はこの不完全な OS を完全に動作させるべく作業を続けた成果であるためです。

ですが、386BSD の大元である 4.3BSD/Net 2 が、AT&T にライセンス違反で提訴され、BSD 自体の開発がストップしてしまいます。この結果、386BSD も公開停止を余儀なくされてしまいます。裁判が CSRG と AT&T との和解に終わり、ようやくライセンス的に問題のない 4.4BSD が公開されたのは 1994 年のことでした。

2.3.3　GNU プロジェクト

1983 年、リチャード・ストールマン (Richard Matthew Stallman) は **GNU プロジェクト**を設立しました [11]。GNU プロジェクトとは、100% 自由ソフトウェアの Unix 互換のシステム (GNU) を提供することを目標とするプロジェクトです [12][*11]。自由ソフトウェアというのは、ソフトウェアの利用者がそのソフトウェアを、ソフトウェア開発者に告知することなく実行、コピー、配布、研究、変更、そして改良する自由を有することを認めるライセンスを持つソフトウェアをいいます [13]。ここで自由ソフトウェアは商用目的を禁止しているわけではありません。つまり、GNU プロジェクトは、かつて AT&T 傘下のベル研究所がそうであったように、誰でも自由に利用できる Unix の環境を取り戻そうとするためのプロジェクトだったのです。続く 1985 年、リチャード・ストールマンは GNU 宣言を発表し[*12]、GNU プロジェクトの理念を明文化し、それに対する支援と参加を求めました。同年 10 月、彼は自由ソフトウェアを推進し、逆に不自由なソフトウェアに反対するために、非営利団体の**フリーソフトウェア財団 (free software foundation, FSF)** を設立し、自由ソフトウェアを強力に推進していくこととなります。

GNU プロジェクトは次々と素晴らしい成果を成し遂げていきました。この業界を Vim と二分するほどに有名なエディタである Emacs やコンパイラの GCC、PGP(Pretty Good Privacy) から派生した暗号化ソフトウェアである GPG などがそれにあたります。ですが、GNU プロジェクトはその目的である Unix 互換の OS である GNU を完成させることはありませんでした。OS に必要な API、ABI、コンパイラ、アセンブラ、リンカなどは既に完成していたのですが、OS の最後のピースを埋めるデバイスドライバやデーモン、そしてカーネルが完成していなかったのです[*13]。

[*10] 以下では特に断りのない限り BSD を OS の名称として使用する。

[*11] GNU は “GNU is Not Unix.” の頭文字をとったもの（再帰的頭字語といわれる）。

[*12] 原文が`http://www.gnu.org/gnu/manifesto.html`で読める（英語）。

[*13] 2001 年に GNU プロジェクトは GNU Hurd というカーネルを完成させたが、このカーネルは現在（2013 年 8 月）に至るまでに実用的なレベルまでには成熟していない。`http://www.gnu.org/software/hurd/hurd.html`を参照。

2.3.4　Linux と GNU/Linux

1987 年、Minix という Unix 互換の OS がオランダ、アムステルダム自由大学 (Vrije Universiteit Amsterdam（オランダ語）) のアンドリュー・タネンバウム教授 (Andrew Stuart Tanenbaum) によって開発されました [15]。Minix はタネンバウム教授が計算機科学の教育に使用するために開発したもので、AT&T のライセンスに抵触しないシンプルな Unix 互換 OS でした。

「Minix はあくまで教育用のホビーであり、実用が目的ではない」[15] というタネンバウム教授の方針のもとで、Minix はクローズドソース[*14] で配布されていましたが、Minix を実用的な OS にしたいと願う人々もユーザの中にはいました。リーナス・トーバルズ（以下リーナス）もその一人でした。

リーナスは、既存の Unix と Unix 互換 OS に不満を抱いていました。Unix は高価で学生身分の彼がおいそれと買えるものではなく、また Unix 互換 OS も、特に彼が使用していた Minix は教育目的に使用が制限されているうえに、その目的によって OS 自身の機能が大幅に簡略化されており、彼が満足するような OS ではありませんでした。そこで、彼は自らの手で OS をつくり上げることにしたのです。1991 年のことでした。

1991 年 8 月 25 日 [9]、リーナスは Minix のユーザネットワークに、「僕は今、 UNIX 互換の OS を作っている」[3] というメッセージを投稿しました。さらに、その OS は自由ソフトウェアであり、いかなる Minix のコードも使用していない [9] と付け加えました。彼の作った Unix 互換の OS こそが、後の Linux カーネルでした。

リーナスの作成したカーネルは、当時の既存の Unix システムに比べ、その機能と実績においてとても比べられるようなものではありませんでした。ですが、既存の Unix システムは自由ソフトウェアではないので自由に公開、再配布が不可能であり、また自由なソフトウェアを目指した BSD は訴訟問題に追われ、GNU プロジェクトは GNU Hurd の開発が難航していたことで OS 開発は途上の段階でした。

つまり、1990 年代初頭に存在した Unix 互換の OS として使えそうなものは、リーナスの開発した Linux カーネル以外を置いて他になかったのです。

リーナスは Linux を GPL ライセンス[*15] の元に公開することを決定し、多数の開発者とともに Linux カーネルの改良にとりかかりました。Linux はあくまで OS のカーネルであり、カーネルのみでは OS として成立させることはできません。Linux を自由ソフトウェアの OS として完成させるための大量の穴は、GNU プロジェクトが Unix 互換 OS である GNU を作るべく開発していたソフトウェア群が埋めてくれました。これは GNU プロジェクトが GNU を作ろうとして完成に至っていなかったカーネルという最後のピースを、Linux が埋めることを意味していました。

かくして、Linux カーネルと GNU プロジェクトのコンポーネントを組み合わせた OS である、GNU/Linux[*16] が誕生したのです。

[*14] プログラムのソースコードを公開しないこと。このようなソフトウェアをクローズドソースソフトウェアといい、逆にプログラムのソースコードを公開するソフトウェアをオープンソースソフトウェアという。

[*15] 自由ソフトウェアに付加するライセンスのうちの一つ。自由ソフトウェアとして実行、コピー、配布、研究、変更、そして改良する自由を有することを認めるが、GPL ライセンスを持つソフトウェアを、あるいはそこから派生したソフトウェアを再配布する場合は、元のソフトウェアと同一の GPL ライセンスに限るというもの。

[*16] Linux は OS のカーネルの名前であり、Linux カーネルを使用し GNU のコンポーネントを使用している OS（これは既存の “Linux” として知られる OS のほぼすべてが該当する）は “GNU/Linux” と呼称すべきだと GNU プロジェクトは主張している（`http://www.gnu.org/gnu/linux-and-gnu.html`などを参照）。この文章でも、既に第 1章でみられるが、この主張に従い、Linux カーネルと GNU のコンポーネントを使用した OS を “GNU/Linux” と呼称する。Linux カーネルを指す場合は呼称として “Linux” 及び “Linux カーネル” の両者を適宜使う。

2.4　Linux ディストリビューション

Linux は OS のカーネルです。Linux カーネルと GNU のコンポーネントを組み合わせて、GNU/Linux と呼ばれる OS システムが出来上がります。このようにして成立する OS は確かに OS ではありますが、私達が普段利用する OS としては機能不十分です。考えてみればわかりますが、GNU/Linux は OS としては完結していますが、OS 上で動かすアプリケーションソフトウェアについてはあまり触れられていません。これは即ち、ブラウザもメーラも、果てはテキストエディタといった、普段私達が使っているようなソフトウェアは GNU/Linux には含まれていません。更にいえば、GUI も GNU/Linux には含まれていません。Linux カーネルと GNU のコンポーネント**のみ**からなる純粋な GNU/Linux を起動した場合、おそらく真っ黒の画面にプロンプトが明滅するインターフェイスが出現し[*17]、用途が非常に限定的な環境が得られるでしょう。このようなデスクトップ用途以外にも、サーバ用途として使う場合、クライアントとの通信の為のソフトウェアやアクセス管理のためのソフトウェア、セキュアな環境を構築するための各種ツール、さらにはサーバが提供するコンテンツのためのプログラムと動作環境など、足りないものは山程あります。

そこで、Linux カーネルと GNU のコンポーネントに加え、利用者がおそらく使うであろうブラウザやメーラ、テキストエディタといった各種アプリケーションソフトウェアを一緒くたにして（バンドルして）配布すると大変便利でしょう。このように、GNU/Linux と各種のアプリケーションソフトウェアをバンドルして配布されているものを**Linux ディストリビューション**といいます。Linux ディストリビューションは OS ではありません。GNU/Linux を要素として含む、ソフトウェアの頒布形態です。例えば一戸建ての家を買うとき、家は土地に付随して販売されます。家は不動産としての資産価値がありますが、その資産価値は住宅と土地を合わせたものであるはずです。Linux ディストリビューションを不動産とすれば、GNU/Linux は土地に該当します。

Linux ディストリビューションには様々な種類があります。駅前徒歩 5 分の一戸建てのような便利なものから、郊外の広い一軒家のようなのびのびと使えるもの、はたまた土地だけ与えられて自分で家を作らされるようなものもあります。興味のある方は [16] をご参照ください。

ちなみに、今回使用する Arch Linux は、残念ながら土地だけ与えられて自分で家を作らされる類の Linux ディストリビューションです。ですが、Arch Linux は既に整地された土地に、手配済みの資材を用いて家を作り上げるような形態ですので、少しは気が楽です。というのは、世の中には荒れ地を自分の手で開墾し、資材も自分で手配することを要求するような Linux ディストリビューション[*18] も存在するのです。

[*17] このように文字だけを介する UI（ユーザインターフェース、User Interface）を CUI(Character UI) という。対してマウスカーソルがあってウィンドウがあってというように、図形的要素を介する UI を GUI(Graphical UI) という。CUI の名誉のために言っておくと、CUI にもブラウザやメーラ、テキストエディタはある。CUI でも Emacs を使えば大抵のことはできる、と言われる。Emacs については`https://www.gnu.org/software/emacs/`を参照。

[*18] `http://www.gentoo.org/`

第 3 章

Arch Linux のインストール

Arch Linux をマシンへインストールし、OSS システムを動作させる機械としてコンピュータを覚醒させます。本ゼミ及びこの文書の中心となる内容です。

3.1 Arch Linux について

Arch Linux は、2001 年にカナダ人プログラマであるジャッド・ビネット (Judd Vinet) により公開され、現在はアメリカ人プログラマであるアーロン・グリフィン (Aaron Griffin) を筆頭に開発を続けている Linux ディストリビューションです [6]。

Arch Linux は、既存のどの Linux ディストリビューションにも依存しない[*1][*2]Linux ディストリビューションとして発展してきました。Arch Linux は、簡潔性、最小性、およびコード緻密性と正確さ (elegance) に焦点を当てて開発されており、このため通常多くの Linux ディストリビューションには大抵付属する GUI による OS のインストーラが付属していません。また、インストール直後のシステムに含まれるパッケージ[*3] の数は同種の Linux ディストリに比べてかなり少なくなっています。これは、お仕着せの環境に縛られず、自分の好きな環境を好きに構成できることを意味し、自由度の高いものと言えるでしょう。

また、Arch Linux は OS 自体に特定のバージョンを定めないローリングリリースというリリース方式を採用しており、システムを常に現状の最新バージョンに保つことができるという利点を持ちます。反面、迂闊にアップデートを行うとシステムが壊れるという危険性も孕んでいますが、問題の解決策は ArchWiki[*4] や Arch Linux Forum[*5] で入手することができます。ArchWiki は膨大な項目数を誇る良質なドキュメントです。困ったときは ArchWiki を読むと良いでしょう。

3.2 Arch Linux イメージの入手

Arch Linux をインストールするには Arch Linux のインストールイメージを入手する必要があります。

まず`https://www.archlinux.org/`にアクセスします。そこから右上の “Download” をクリックし、Arch Linux

[*1] 例えば Ubuntu は Debian から派生し、Fedora は RHEL の開発版的位置付けである。同じく CentOS は RHEL が公開しているソースコードを元に開発されたものである。このように、既存のディストリ（Linux ディストリビューション）から派生し、あるいは関連して新たなディストリが発生することも多い。

[*2] このような方針をとるディストリは他に Gentoo、Slackware があるが、いずれも荒野を開墾していく系のディストリである。独自の方針をとるディストリは並べてどうも泥遊びが好きらしい。

[*3] システムにインストールするプログラム群の最小単位。インストールするプログラム自身と、パッケージの管理のために使用するメタデータを含んでいる。

[*4] `https://wiki.archlinux.org/`

[*5] `https://bbs.archlinux.org/`

Downloads のページに移動します。Arch Linux のインストールイメージには図 3.1のような命名規則が定められています。

archlinux-<西暦>.<リリースされた月>.<バージョン番号>-dual.iso

図 3.1: Arch Linux インストールイメージの命名規則

このうちの「<西暦>.<リリースされた月>.<バージョン番号>」は Arch Linux Downloads 内の “Current Release” の内容と一致するので、どのバージョンが利用可能なのかがここで確認できます。

ダウンロードに際し、BitTorrent が利用できる環境であれば “recommended” に従い torrent ファイルをダウンロードしてイメージを入手するのが良いでしょう。BitTorrent が利用できない場合は、“HTTP Direct Downloads” から日本のサーバを探し、ダウンロードします。この文書を執筆している 2014 年 5 月 28 日の段階では、`ftp.tsukuba.wide.ad.jp` と `jaist.ac.jp` の 2 箇所が利用できます。

ダウンロードしたファイル名は、デフォルトでは命名規則に従ったものとなっています。これを適当なディレクトリに保存します。可能であれば、“HTTP Direct Downloads” の “checksum” よりイメージのハッシュ値を入手し、ダウンロードしたファイルが正当なものかを確認するとよろしいでしょう。

3.3　Arch Linux イメージの起動

さて、Arch Linux イメージを入手ました。手に入れた道具は使わなければ宝の持ち腐れです。早速 Arch Linux を起動してみましょう。イメージを CD に焼いて光学ドライブ経由で起動するなり、仮想マシン上に起動メディアとして Arch Linux イメージを指定して起動するなり、適当な方法で起動して下さい。

さて、順調に起動処理が進むと、図 3.2のような画面が表示されます。これはブートローダが OS をブートする過程で、起動する OS をユーザが選択するフェーズです。今回仮想マシンにインストールするのは 64 ビット版の Arch Linux ですので、“Boot Arch Linux (x86_64)” を選択し、Enter キーを押下します。

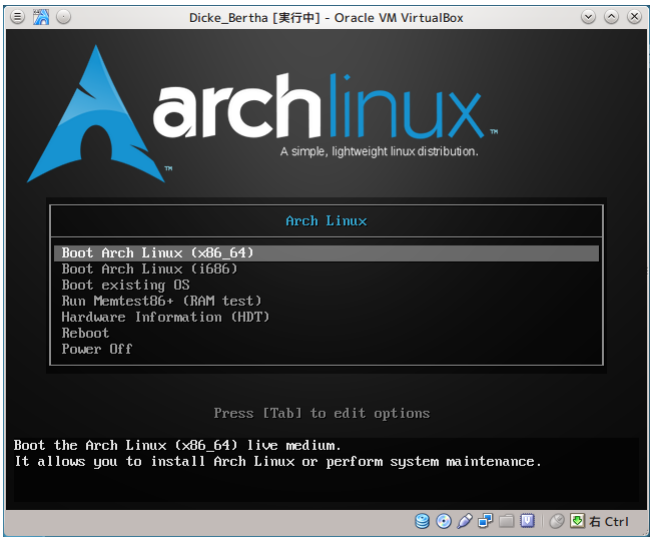

図 3.2: 起動する OS を選択する（ブートローダ）

OS を選択後、しばらくすると図 3.3のような画面が表示されます。これが Arch Linux です。

以降では、このようにキーボードでコマンドを逐次叩いていく CUI によって、OS をインストールしていきます。

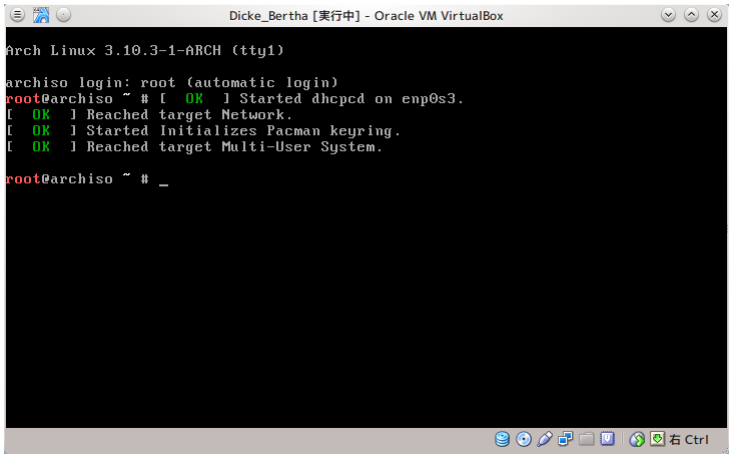

図 3.3: Arch Linux の起動

3.4　Arch Linux をインストールする

それでは早速インストールを始めていきましょう。
インストールの基本的な流れは以下の通りです。

1. インストールイメージを起動する
2. キーマップを変更する
3. ネットワークへの接続を確認する
4. ディスクを必要な領域ごとに切り分ける（パーティションを作成する）
5. 4. で作成したパーティションにファイルシステムを作成する
6. 5. で作成したファイルシステムをマウントし、OS のプログラムをディスクへ流し込む
7. ディスク上の OS の設定を行う
8. root アカウントのパスワードを設定する
9. ブートローダをインストールする
10. システムを再起動させてディスクから起動し、root アカウントでログイン、動作を確認する

現在までに 1. が終了しました。以下では順を追って作業を進めていきます。

ここで、本文中ではコマンドを扱いますが、コマンド名は本文中ではタイプライタ体（`typewriter type`）で表現し、実際にコマンドを用いての作業内容はタイプライタ体文字列を影付き枠で囲うことで表します。必要があれば、コマンド実行中の画面あるいは画面の一部を二重枠で囲うことで表します。ここでコマンドを記述する行の先頭に “`$`” や “`#`” が書かれる場合がありますが、これは実際の作業画面に於けるプロンプトを再現したものであり、実際には入力の必要はありません。加えて、斜体 (*`Italic`*) で表現されている箇所は字体に関わらず各位による変更を必要とする箇所を示します。また本文中では、「コマンドを入力する」という行為を簡略のため「叩く」と表現します。

本文中で扱われるコマンドについては適宜必要と思われる部分に関する説明を加えていきますが、いまいちピンとこない場合などは、man を用いることでコマンドの詳細な説明を得ることができます。

```
$ man command_name
```

と叩くと `command_name` という名前のコマンドに関する説明を閲覧できます。

3.4.1　キーマップを変更する

Arch Linux のデフォルトのキー配列は US 仕様になっています。ですが日本語を第一言語とする方々の大多数は JIS 配列のキーボードを使用しているはずです[*6]。このため、キー配列を変更しないと、例えばダブルクォーテーション (“) を入力したいのにアットマーク (@) が入力されたり、括弧を閉じたいのに閉じなかったりと非常にイライラするので、キー配列を JIS 配列に変更します。

```
# loadkeys jp106
```

と叩き、キー配列を変更しましょう。

3.4.2　ネットワークへの接続を確認する

Arch Linux のインストールイメージにはインストールする OS のプログラムは実は入っていません。インストールイメージは OS という家を建てるための足場のようなものです。OS のプログラムは、Arch Linux のサーバから持ってこなければなりません。よって、インターネットへの接続が確立しているか確かめる必要があります。

固定 IP アドレスを割り当ててインターネットへ接続している等、特に特殊な環境でない限り、Arch Linux のインストールイメージは `dhcpcd` というネットワークデーモン[*7] により、既にインターネットへ接続されています。ここでは接続が確立していることを確かめるにとどめます。

```
# ping -c 3 www.tmu.ac.jp
```

と叩き、接続を確認します。`ping` とは引数に指定したサーバへ少量のネットワークパケットを送信してサーバとの通信状態を確認するコマンドです。今回はオプションに “`-c 3`” を加えて、パケットの転送回数を 3 回に制限しました。

3.4.3　ディスクを必要な領域ごとに切り分ける（パーティションを作成する）

ディスクの確認

続いて、ディスク上にパーティションを作成していきます。パーティションとはディスクの記憶領域が分割されたものです。例えば 10GB のディスクを 2GB と 8GB に分割したとすれば、そのディスクは 2GB のパーティションと 8GB のパーティションを持つことになります。

パーティションの作成はこだわり出すと止まらなくなることがままあるので、ここはベターな作成法として、ディスクを分割せずに、ディスクをまるごと 1 つのパーティションとして扱うことにします。

通常仮想マシンのディスクはインストールイメージには “/dev/sda” として認識されています。後述しますが、“/dev” とはコンピュータに接続されたデバイス (device) を管理するディレクトリ[*8] で、sda とはシステムに最初に認識されたディスクであるという意味です。今扱っている仮想マシンではディスクは 1 個のみですので、システムでは “/dev/sda” と認識されています。確認したい場合は、

```
# lsblk
```

と叩き、sdX(X は a–z の間の任意の一字。通常は a) の行を探しましょう。通常は先頭に表示されているはずです。

[*6] US 配列のキーボードを使用している方はこの小節は読み飛ばして構わない。
[*7] デーモンとは、システム上で動いているプロセスのうち、表に出てこない（バックグラウンドで実行されている）ものを指す。
[*8] 別称フォルダ。今後はいわゆるフォルダもディレクトリと呼称する。

発見したら、その行の “SIZE” の列を確認し、インストール先のディスクの容量と一致しているかを確かめると良いでしょう。

パーティションの作成

いよいよパーティションを作成します。以下では、ディスクが “/dev/sda” と認識されていることを仮定します。

ディスクにおけるパーティションの情報は、パーティションテーブルとよばれるもので管理されています。現在、このパーティションテーブルの方式には MBR(master boot record) と GPT(GUID partition table) との 2 種類があります [18]。MBR は枯れた方式で、GPT はモダンな方式です。ここでは新しさを求めて GPT でのパーティション管理を試みます。

まず以下を叩きます。`cgdisk` とは、ディスク上に GPT パーティションを作成するためのユーティリティです。

```
# cgdisk /dev/sda
```

すると図 3.4のような表示になるはずです。まれに “Warning! Non-GPT or damaged disk detected!” などと怒られる場合がありますが、どうせ新たにパーティションを作るのですからシカトして Enter を押下しましょう。

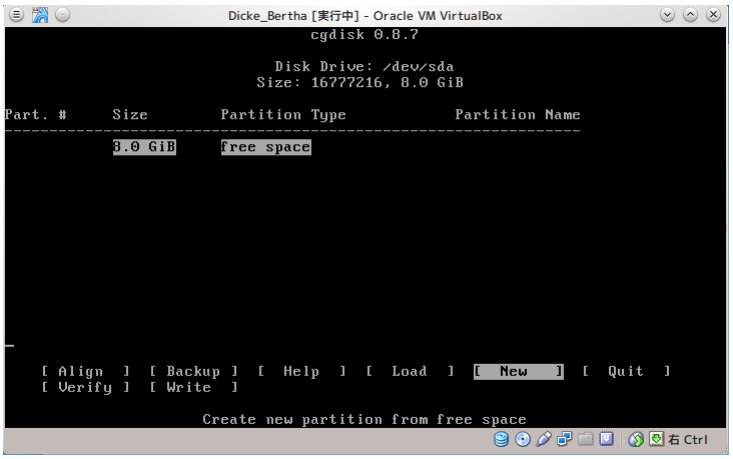

図 3.4: cgdisk

まず “New” を選択（左右矢印キーで選択可）するか N キーを押下してパーティションの作成に入ります。あとは元の表示になるまでずっと Enter キーを叩けば良いのですが、折角ですので尋ねられる文章の意味を記しておきます。

- First Sector (XXXX–YYYY, default=XXXX)（XXXX、YYYY は適当な値）
  - パーティションの開始位置を指定する。XXXX はディスク先頭 0 バイトから XXXX バイト目を示す。
- Size in Sectors or {KMGTP} (default=XXXX)（XXXX、YYYY は適当な値）
  - パーティションのサイズを指定する。デフォルト値 XXXX は今作成しているパーティションの後ろに別のパーティションが存在しないため、ディスクのサイズに等しくなっている。
    セクタ (Sector) というのは容量の単位。KMGTP も同様 (KiB、MiB、GiB、TiB、PiB)[*9]
- Current type is 8300(Linux filesystem)
  Hex code or GUID (L to show codes, Enter = 8300)
  - パーティションのタイプを指定している。Linux 用パーティションの場合 8300 が該当する。

[*9] 2 バイト接頭辞。それぞれ 2^{10}B、2^{20}B、2^{30}B、2^{40}B、2^{50}B を意味する。

- Current partition name is ”
  Enter new partition name, or <Enter> to use the current name:
  – パーティション名を決める。デフォルトでは名無しのパーティションとなる。

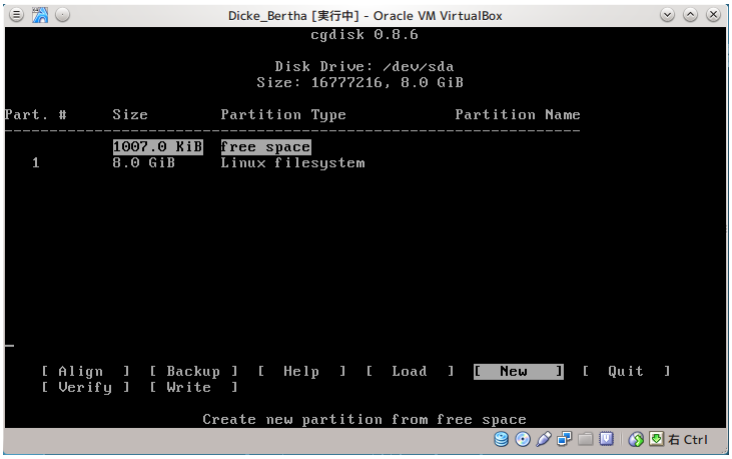

図 3.5: cgdisk（パーティション作成後）

もとの表示（図 3.5）に戻ったら、まず作成したパーティションを上下矢印キーを用いて選択し、次に左右矢印キーを用いて “Write” を選択するか Shift キーを押しながら W キーを押して[*10] パーティションをディスクへ書き込む作業へ入ります。“Areyou sure you want to write the partition table to disk? (yes or no):” と聞かれるので、yes と入力し（“y” でないことに注意）パーティションをディスクに書き込みます。書き込みが終了したら再び図 3.5の表示に戻るので、左右矢印キーを用いて “Quit” を選択するか Q キーを押して cgdisk を終了します。

パーティションの作成の確認には `lsblk` を使用すると良いでしょう。

```
# lsblk
```

を叩き、“sda” を確認すると、その直下に “sda1” が新たに表示されているはずです。この行の “TYPE” を確認すると、“PART” となっていると思います。これは PARTition の意で、sdX1 はディスク sdX に作成されたパーティションを意味しています。ちなみに、sda にパーティションを何個も作っていくと、もちろん sda2、sda3、. . . と認識されていきます。

3.4.4　ファイルシステムの作成

以上までの作業で、OS という家を建てる上での地均しは終了しました。次に基礎を築きます。

今回はパーティションを ext4(Fourth Extended File System) というファイルシステムでフォーマットします。ファイルシステムとは、ディスク上に存在するファイルを管理するためのシステムで、OS の機能の一部です。

/dev/sda1 を ext4 でフォーマットするには、以下を叩きます[*11]。

```
# mkfs -t ext4 /dev/sda1
```

以後の説明では、適宜「ディスク」と「ファイルシステム」を同義なものとして扱います。ハードウェアとしての

[*10] 以後このようなキー操作を “Shift+W” のように表現する。
[*11] [4] では `mkfs.ext4 /dev/sda1` となっているが、この文書でのコマンドの用法の統一のため異なる形式を用いた。

ディスクを扱う場合、文脈上に説明を加えることとします。

3.4.5　ファイルシステムのマウント

ファイルとディレクトリ階層

マウントとは、コンピュータに接続されたデバイスを、OS が管理および制御できる状態にすることを指します。ファイルシステムをマウントする場合、Linux を始めとする Unix 互換 OS では `mount` というコマンドを用います。

元来、Linux には “すべてのものはファイルである (everything-is-a-file)” という思想があります [1]。つまり、ディスクや CD ドライブ、ディスプレイやプリンタやキーボード、はたまたディレクトリや通常の意味でのファイル、さらにプロセスに至るまでの全てが Linux においてはファイルとして扱われます。アクセスの方法も道具の相異は有りますがほぼ統一されており、それは単純に読み書きとして表現できます。

さて、Linux のファイルは “/” で表現されるルートディレクトリを頂上に、全てのファイルとディレクトリは “/” の下に位置します。厳密にはファイルの目的に応じて、“/” 直下にいくつかのディレクトリが存在し、ファイルとディレクトリは “/” 直下のディレクトリに振り分けられて存在しています。Linux のとるこのようなディレクトリ階層は FHS(Fileystem Hierarchy Standaed) と呼ばれ、標準化されていますが、実際の Linux ディストリビューション同士では細かい差異が存在しています。

Arch Linux におけるディレクトリ階層は図 3.6のようになります [17][18]。

ファイルシステムのマウント

[4] によれば、OS をインストールするディスクを/mnt 下にマウントしています。この文書に於いても [4] に従い、/mnt 下にマウントすることにしましょう。また、今回はディスク上にはファイルシステムが 1 つしか存在しませんので、そのファイルシステムを “/” として扱い（以下 “/ファイルシステム” と呼称します。）/mnt にマウントします。ちなみに、ファイルシステムは通常データを階層的に管理するために扱われるため、ファイルシステムをマウントするときはディレクトリにマウントしないとエラーが起きます。

```
# mount /dev/sda1 /mnt
```

と叩き、ファイルシステムをマウントします。注意すべきことして、マウントするのはディスクではなくファイルシステムですので、`mount` に指定する引数は/dev/sda ではなく/dev/sda1 です。

3.4.6　OS をインストールする

いよいよ OS をインストールします。

OS をインストールする前に、まずは OS のインストールに必要なパッケージをダウンロードするサーバを決めます。

OS を構成するパッケージは、必要なものを全てひっくるめると数百 MB もの大きさになり、パッケージ群を提供するサーバ及び通信回線に多大な負荷を及ぼします。更に、このパッケージ群を必要とするクライアントは何百何千、何万何十万という数になりますから、これを 1 台のサーバで賄おうとした場合サーバは通常処理能力の限界を瞬時に迎え即死します。このため、通常大規模なデータを多数のクライアントから要求されるような状況で扱う場合は、負荷分散のためミラーサーバと呼ばれるサーバを複数台用意し、特定のサーバに負荷が集中しないようにしています。

サーバを分散することで得するのは運営側のみではありません。同じデータをダウンロードするにしても、例えば日本にいる利用者が日本にあるサーバからダウンロードするのとブラジルにあるサーバからダウンロードするのとでは掛かる時間は雲泥の差です。回線の性能や使用状況等による差異も有りますが、同程度の回線を使用しているならば、物理的な距離の近い経路からアクセスしたほうが、一般にダウンロードに要する時間は少なく済むであろうことは直感的

/	root ディレクトリ（システムの最上位ディレクトリ）
├ bin	全てのユーザ用の基本コマンドを格納するディレクトリ
│	（現在は/usr/bin へのシンボリックリンク）
├ boot	ブートローダ用ファイルとカーネルイメージを格納するディレクトリ
├ dev	デバイスファイルを格納するディレクトリ
├ etc	OS の設定ファイルを格納するディレクトリ
├ home	一般ユーザ用ディレクトリ
├ lib	共有ライブラリとカーネルモジュールを格納するディレクトリ
│	（現在は/usr/lib へのシンボリックリンク）
├ (lib64)	64bit 環境用の共有ライブラリを格納するディレクトリ
│	（現在は/usr/lib へのシンボリックリンク）
├ lost+found	ファイルシステム障害発生時に復旧可能なファイルの一時保管場所
├ mnt	ファイルシステムの一時的なマウントポイントを格納するディレクトリ
├ opt	通常の Linux のディレクトリ階層では扱えないプログラムを格納するディレクトリ
├ proc	カーネル及びプロセスの情報を格納するディレクトリ
├ root	root ユーザ用ディレクトリ
├ run	短期的に使用されるファイルを格納するディレクトリ
├ sbin	システム管理用コマンド等を格納するディレクトリ
│	（現在は/usr/bin へのシンボリックリンク）
├ srv	HTTP、FTP 等のサービス用のデータ
├ sys	OS 全体の情報を格納するディレクトリ
├ tmp	一時的なファイルを格納するディレクトリ
├ usr	各種プログラム等を格納するディレクトリ
└ var	頻繁に変更されるデータを格納するディレクトリ

図 3.6: Arch Linux のディレクトリ階層

にも納得できると思います。

Arch Linux は現在世界各国にミラーサーバを構え、日本国内にも北陸先端科学技術大学院大学（通称 JAIST) が提供するものと、筑波大学が提供するものとの 2 種類があります。察しがつくと思いますが、インストールイメージを入手する際に説明した 2 つのサーバである `jaist.ac.jp` と `ftp.tsukuba.wide.ad.jp` がそれぞれ該当します。どのミラーサーバにアクセスするかを設定するには/etc/pacman.d/mirrorlist を編集します。どちらのサーバを選んでも良いのですが、今回は JAIST のサーバを使う方法でいきたいと思います。筑波大学のサーバを使う場合も、以後で解説する方法と同様に行うことが可能です。

ミラーサーバの設定

/etc/pacman.d/mirrorlist を設定します。

このファイルは前述の通り、パッケージのダウンロードにどのサーバを使うかを指定するための設定ファイルです。Linux の設定ファイルはほぼ全てテキストファイルですので、設定を変更したりする場合にはテキストエディタを使用します。今回は一般的なテキストエディタに近い使用感を持つ `nano` というエディタを使用します。以後の設定ファイルの編集に於いても `nano` を使用していきます。

```
# nano /etc/pacman.d/mirrorlist
```

と叩くと、図 3.7のような表示になります。これが nano のインターフェースです。

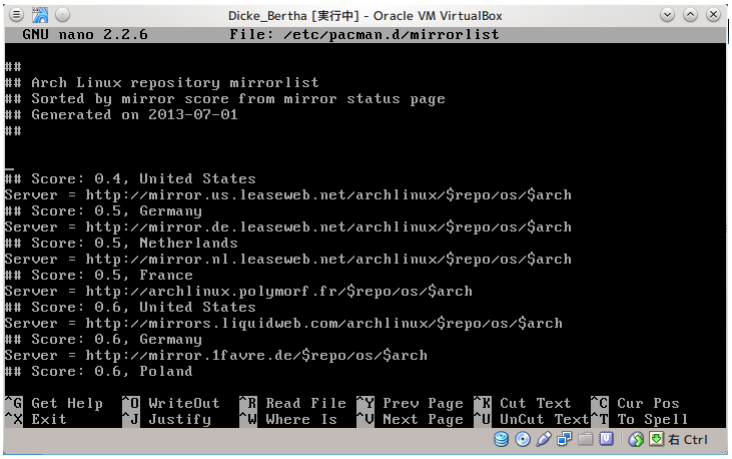

図 3.7: nano による/etc/pacman.d/mirrorlist の編集

このファイルには世界に点在する Arch Linux のミラーサーバの URL が収録されています。このまま何も変更も加えずにパッケージのダウンロードを試みると、有効なサーバとして mirror.us.leaseweb.net が選択、使用されます。これはアメリカに存在するサーバです。行頭に “#” がついている行はコメントアウトされた行といい、データとして意味を成さない記述となります。文字通り「コメント」を書き込む際に用いられるものです。従って、コメントアウトされていない行のうち最も上位にある mirror.us.leaseweb.net が選択されます。

即ち、mirror.us.leaseweb.net よりも上に JAIST のサーバの URL を書き込めば、パッケージのダウンロードは JAIST のサーバより行われることとなります。では早速 JAIST のサーバの URL を探し出しましょう。

まず Ctrl+W を入力してファイル内の文字列検索へ移行します。

```
Search:
```

という表示が画面下部に現れますので、そこへ “japan” と入力し、Enter キーを押下します。するとファイル内で “Japan” の文字列が含まれる行へカーソルが移動します。該当するものが複数個存在する場合、現在のカーソル位置から最も近い行へ移動します。/etc/pacman.d/mirrorlist では JAIST のサーバと筑波大学のサーバと両方が含まれていますが、JAIST のサーバのほうがファイル内で上位に記載されているので、通常は JAIST のサーバの URL 付近にカーソルが移動するはずです。

続いて JAIST サーバの URL へ上下矢印キーでカーソルを合わせたら、Alt+6 でカーソルの存在する行全体をコピーします。次に PgUp キーでファイルの上方へ移動し、mirror.us.leaseweb.net よりも上の行にコピーした URL を貼り付けます。コピーした内容の貼付けには Ctrl+U とします。

図 3.8のように編集が完了したら、ファイルを保存しエディタを閉じます。まず Ctrl+O を入力し、ファイルを保存します。このとき、

```
File Nameto Write:  /etc/pacman.d/mirrorlist
```

と表示されます。コロン以降を変更すればファイル名の変更が可能ですが、今回はそのまま Enter を押下します。すると

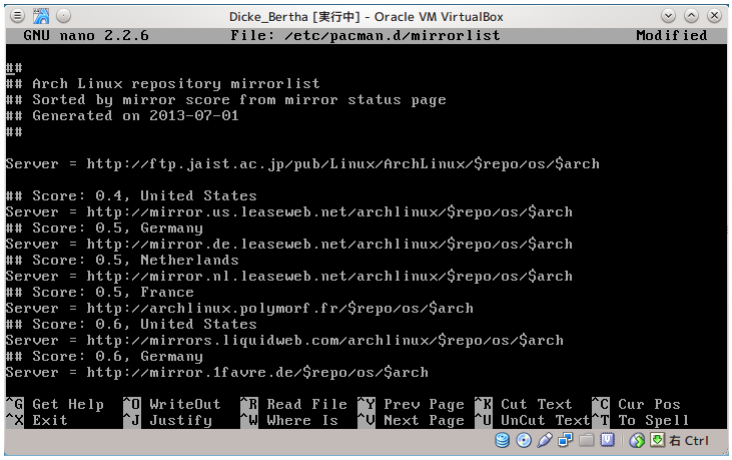

図 3.8: 編集後の/etc/pacman.d/mirrorlist

```
Wrote XXX lines
```

と表示され、ファイルが保存されます（XXX はファイルの行数を示す整数）。続いて、Ctrl+X を入力し、nano を終了します。

パッケージのダウンロード

いよいよ正真正銘の OS のインストールです。

OS インストールには pacstrap というスクリプトを使用します。これは後述する pacman のラッパ[*12] であり、このスクリプトを使うことで、OS のインストール先を指定してやるだけでほぼ何も考えずに OS のインストールが自動で成されるというスグレモノなのです。

```
# pacstrap -i /mnt base
```

と叩き、インストールを開始します。コマンドの意味は、/mnt に base カテゴリのパッケージのみをインストールする、というものです。base カテゴリにはこれさえあれば最低限は使えるというパッケージが含まれています[*13]。

/etc/fstab の編集

OS のインストールも折り返し地点に近づいてきました。続いて/etc/fstab の編集に移ります。

OS 起動後、OS はシステムが格納されているファイルシステムをマウントしなければなりません。（詳しくはファイルシステム内に OS が存在するため、OS 起動前にファイルシステムは認識できません。このあたりの詳細は後述します）OS 起動後に OS がマウントすべきファイルシステムを記載した設定ファイルが/etc/fstab です。これが正しく設定されていないと OS は起動時にエラーを出力し、起動に失敗します。

/etc/fstab を編集する前に、まずは/etc/fstab を作成しましょう。以下を叩きます。

```
# genfstab -L -p /mnt >> /mnt/etc/fstab
```

genfstab で/mnt にマウントされているファイルシステムにおける fstab を作成し、それを/mnt にマウントされて

*12 ある関数を使うために、関数および関数に必要なデータや引数、別の関数などを一緒にまとめて定義した関数。ここでは「関数」を「コマンド」に置き換えたものが該当する。wrapper。

*13 私達が通常想起する「最低限」とは程遠いことに注意。もっと低い次元の「最低限」である。

いるファイルシステムに書き込みます。

続いて/etc/fstab を編集していきます。

```
# nano /mnt/etc/fstab
```

と叩くと、以下のような表示がなされると思います。

```
#
# /etc/fstab:  static file system information
#
#<file system> <dir>  <type>   <options>                  <dump> <pass>
# /dev/sda1 UUID=d9a74548-5c57-23c6-91f2-888a1bc2fd67
/dev/sda1      /      ext4     rw,relatime,data=ordered   0      1
```

チェックすべきは一箇所で、“<pass>” の列の値が “1” となっているかを確認してください。他は特にいじらなくて良いと思います。

3.4.7　OS の設定を行う

OS のインストール自体は完了しました。以後は OS の設定やユーザの追加、各種パッケージの追加、ブートローダのインストールを行います。

chroot(arch-chroot)

現在使用している Arch Linux という OS は言うまでもなくインストールメディアとしての Arch Linux であり、ディスクにインストールした Arch Linux ではありません。ディスク上にインストールされた OS の設定を行うには、やはり OS をインストールしたディスク内の設定ファイルを変更しなければなりません。ですが、今の状態ではファイルシステムが/mnt にマウントされている都合上、設定ファイルのパスの先頭に “/mnt” を付けなくてはならず、面倒です。加えて、ディスク上の OS でコマンドを実行したい場合も、同じくマウントしたファイルシステム経由ではいささか不都合が生じます。

というわけで、ディスク上の OS を操作できるようにしましょう。これには、`chroot` のラッパである `arch-chroot` というスクリプトを用います。これは root ファイルシステムを変更 (change root filesystem) するためのものです。

```
# arch-chroot /mnt
```

と叩くことで、/mnt にマウントされたファイルシステムを root ファイルシステムとして扱います。これはディスク上にインストールされた OS を起動している状態と同等です。

キーマップの変更

環境が変更されたのでキー配列もデフォルトの US 仕様になっています。

```
# loadkeys jp106
```

と叩き、キー配列を変更しましょう。

また、これは一時的な変更であり、インストール完了後システムを再起動すると設定は無効化され、キー配列は再び US 配列に戻ってしまいます。これを阻止するには、/etc/vconsole.conf に所定の設定を記述する必要があります。

```
# nano /etc/vconsole.conf
```

を実行してファイルを開き、そこへ

```
KEYMAP='jp106'
```

と記述します。

なお、これはコンソール上、即ち CUI 上のみで有効な設定です。GUI に於いては設定が反映されない場合があります。GUI 上におけるキーマップの設定は、第 4章で扱います。

ロケールの設定

ロケール (Locale) とは、ユーザが扱う言語を規定するものです。この分野の標準語は事実上英語であり、デフォルトでは Arch Linux も英語が使用されていますが、ロケールを変更することにより、英語以外の言語を使用することができます。用意されている言語はヨーロッパ方面に限らず全世界にまたがっています。

ですが、ここでは扱う言語は英語のままとします。というのも、ASCII に含まれない文字コードを扱う場合、特に何も設定していない CUI 上では文字化けを起こすためです。残念ながら我々の日本語は ASCII には含まれない文字コードを扱うので、使えません。では日本語は使えないのかというとそういうわけではなく、CUI では英語、GUI では日本語というように設定することが可能です。詳細は第 4章で扱います。

ではロケールの設定をしましょう。最初に、使用するロケールを設定します。これは/etc/locale.gen によって設定されています。これを編集します。以下のように叩きます。

```
# nano /etc/locale.gen
```

ロケール名は、“xx_yy.ZZZ” というフォーマットになっています。xx は言語コード、yy は国名コード、ZZZ は文字コードをそれぞれ示します。例えばロケール名が “en_US.UTF-8” である場合、このロケールは文字コードに UTF-8 を用いるアメリカ英語であるという意味になります。

先程「CUI では英語、GUI では日本語というように設定することが可能」と説明しましたが、/etc/locale.gen が扱う内容はシステムすべてのロケールであり、ここで日本語を扱うように設定しないと、システム上で日本語を扱うことが不可能になります。ちなみにロケールは複数選択が可能です。

ここでは以下のロケールをアンコメント[*14] します。

- en_US.UTF-8
- en_US.ISO-8859-1
- ja_JP.EUC-JP
- ja_JP.UTF-8

設定ファイルを編集したら、今度は設定を反映させます。

```
# locale-gen
```

と叩くことで設定が反映されます。

続いて、CUI 上で使用する言語を規定します。これは/etc/locale.conf で設定されます。今回は英語を使用するので、/etc/locale.conf には

[*14] コメント行の先頭にある “#” を削除すること。これにより記述が設定として反映される。

```
LANG=''en_US.UTF-8''
```

とだけ書いて保存します。

タイムゾーンの変更

続いてシステムの時間を設定します。インターネットに接続していれば、時刻サーバと標準時が自動的に同期されるため、正しいタイムゾーンを設定すれば時刻は自動的に正しい時刻に補正されます。

Linux では日本標準時はおおよその場合 “Asia/Tokyo” として表現されます。タイムゾーンの情報は/usr/share/zoneinfo/下に置かれています。日本のタイムゾーンの情報は、/usr/share/zoneinfo/Asia/Tokyo となります。これを/etc 下に置きますが、ファイルのコピーを置くのではなく/usr/share/zoneinfo/Asia/Tokyo へのシンボリックリンク[*15] を/etc に置くようにします。シンボリックリンクの名称は/etc/localtime とします。

```
# ln -s /usr/share/zoneinfo/Asia/Tokyo /etc/localtime
```

ハードウェア時刻の変更

次にハードウェアの時刻を設定します。適切なハードウェア時刻を設定しないと、システムの時刻に誤差が生じる可能性があります。

[4] によれば、ハードウェア時刻には UTC[*16] を設定することが推奨されているので、以下のように叩きます。

```
# hwclock --systohc --utc
```

ホスト名の変更

ホスト名とはいわばコンピュータの名前です。ネットワーク上でコンピュータを指し示すものとして扱われます。

設定自体は/etc/hostname というファイルに記載された文字列がホスト名として認識されますが、今回は引数をそのまま標準出力[*17] に表示する `echo` というコマンドと、得られた出力を入力として別の処理へ与えるリダイレクトという機能を用いて簡潔に設定することにします。

```
# echo ‘‘HOSTNAME’’ > /etc/hostname
```

と叩くことで、ホスト名が *HOSTNAME* に設定されます。詳細を説明すると、まず echo によって *HOSTNAME* は標準出力に出力されますが、リダイレクトが使用されているので、*HOSTNAME* はそのままリダイレクト先への入力として扱われます。リダイレクトは入力文字列として HOSTNAME を受け取り、そのまま出力先に指定された/etc/hostname に内容を書き込みます。

処理結果を確認するには、エディタを用いるか、`cat` というコマンドを用いて

```
# cat /etc/hostname
```

とすれば良いでしょう。`cat` は引数に指定したファイルを標準出力上に表示するコマンドです。

[*15] あるファイルへのアクセスを異なる場所から行うための手段。ここでは/usr/share/zoneinfo/Asia/Tokyo へのアクセスにそれを直接指定するのではなく/etc/localtime を指定することで/usr/share/zoneinfo/Asia/Tokyo へアクセスが可能となる。

[*16] 協定世界時 (coordinated universal time)。標準時の基準となるもの。

[*17] 通常はディスプレイを意味する。

ネットワークの設定

Arch Linux では、ネットワーク管理の為のネットワークマネージャとして、規定としては netctl を使用します [19]。netctl は CUI ベースのネットワークマネージャであり、GUI を用いたネットワークマネージャのような直感的な使い勝手はありませんが、テキストファイルから構成される簡潔な設定ファイルを用いることで柔軟なネットワーク設定を行うことが出来ます。

netctl は chroot された環境では動作出来ません。ここでは、以下を叩くことで netctl を動作させる為の依存関係を解決するにとどめておきましょう。netctl については第 4章で扱います。

```
# pacman -S dhclient
```

ネットワークへの接続にイーサネットによる有線接続でなく wifi による無線接続を用いる場合は、こちらも依存関係を満たすために wpa_supplicant というパッケージをインストールする必要があります。当該環境においては、上記の代わりに下記を叩いてください。

```
# pacman -S dhclient wpa_supplicant
```

3.4.8　root アカウントのパスワードを決定する

OS を使用するユーザのアカウントには通常一般ユーザとスーパーユーザの 2 種類が存在します[*18]。一般ユーザというのは、字面の通り通常用途に使用するユーザのアカウントであり、OS 内で行使できる権利に制限があります。例えばシステム全体に影響する設定の変更は一般ユーザには許可されていません。対して、スーパーユーザというのは OS 内で行使できる権利に制限のないアカウントです。スーパーユーザの権限は実に強力で、スーパーユーザが定められた OS の中であれば全ての行為が実現できます[*19]。いわばそれはシステムを創造することもでき、またシステムを破壊することもできるのです。ちなみに、スーパーユーザのアカウントは Linux では通常 root アカウントと呼ばれています。

デフォルトでは root アカウントのみがシステム上には存在しています。よって、通常は一般ユーザ用のアカウントも新たに作成するのですが、それはインストール作業が完了してから、ということにしましょう。デフォルトの root アカウントにはパスワードが設定されていません。まずはこちらを先にどうにかしましょう。

```
# passwd
```

と叩くと、

```
Enter new UNIX password:
```

というプロンプトが表示されるので、パスワードを入力します。ここで、キーボードをいくら叩こうとも、入力内容は画面に表示されないことに注意してください。画面に表示されなくとも入力された内容は保持されていますし、ミスタイプした場合は Backspace キーを使用して訂正することも可能です。

パスワードを入力したら、

```
Retype new UNIX password:
```

[*18] 厳密にはこの 2 種類に加え、特定のプロセスが使用するシステムアカウントと呼ばれる 1 種類が加わる。

[*19] 実はできないこともある。例えば、如何なるユーザ、グループにも読み書き実行を許可しないファイルへの読み書き実行はスーパーユーザでも不可能。スーパーユーザは全能の存在ではない。

と表示されますので、再度同じパスワードを入力します。
入力に成功したら、

```
passwd:  password updated successfully
```

と表示されます。

3.4.9　ブートローダをインストールする

OS は通常ディスク内（詳しくはディスク上のファイルシステム内）に存在するということは既にご存知であると思います。OS がファイルシステムを管理下に置くには、ファイルシステムをマウントしなければなりません。さて、ここで疑問が生じます。ファイルシステム内に存在する OS をコンピュータ起動時に呼び出すにはどうすればいいのでしょう。コンピュータの起動時には当然 OS は動作していません。すると OS が起動していないのでハードウェアを制御下に置けません。しかし OS はディスクというハードウェアの中に存在するプログラムです。困ったことになりました。OS を起動するためには OS が必要ですが、OS を起動するための OS はどのように起動すればよいのでしょう[*20]。
これを解決するのがブートローダ（ブートストラップローダ)[*21] というプログラムで、実はこのブートローダがいわば「OS を起動するための OS」となるのです。
では早速導入していきましょう。今回は syslinux というブートローダを使用します。

```
# pacman -S syslinux gptfdisk
```

と叩き、syslinux のパッケージを導入します。続いて、ブートローダの設定のために、

```
# syslinux-install_update -iam
```

と叩きます。最後に、syslinux の設定ファイルである/boot/syslinux/syslinux.cfg を設定します。
デフォルトではカーネルイメージの場所が/dev/sda3 となっているのですが、もちろん/dev/sda3 というパーティションは存在しないので、これを/dev/sda1 に書き換えます。/dev/sda3 となっている箇所が複数箇所ある場合は、その都度/dev/sda1 に書き換えます。設定を終えたら変更を保存して終了します。

3.4.10　インストールの終了

これでインストールの全工程が終了しました。お疲れ様でした。最後にシステムを終了し、仮想マシンをオフにします。
まずシェル[*22] 上で

```
# exit
```

と叩くと `chroot` が終了し、インストールメディアへ帰ってきます。`exit` は本来ユーザをログアウトするためのコマンドですが、`chroot` 下においては `chroot` の終了を意味します。

[*20] 似たような問題に「コンパイラをコンパイルするためのコンパイラのコンパイルはどうすればいいのか」というものがあり、これはブートストラップ問題と呼ばれる。「服を買いに行くための服が無い」という問題との類似点が指摘されている［要出典］。ちなみにコンパイラを処理するためのコンパイラのコンパイルは元を辿って行くと最初期には手で行われた（はず）し、そもそもソフトウェアではなかった。
[*21] 英語では bootstrap loader と書く。ブーツのつまみ革 (bootstrap) に由来する。ブーツのつまみ革ってなんだという人は、小学生が履いている上履きのかかと部分についてたアレを思い出すと良いかもしれない。
[*22] 今までコマンドを入力してきた環境をシェルという。現在使用しているシェルは CUI のシェルであり、GUI のシェルも存在する。

続いて、マウントしていたファイルシステムをアンマウント[*23] します。

```
# umount /mnt
```

と叩くことで、/mnt にマウントされていたファイルシステムをアンマウントします。
最後にシステムをシャットダウンしましょう。これには以下のように叩きます。

```
# shutdown -h now
```

字面の通り、現時点を以って電源を落とせ、という意味です。これにより OS はシャットダウン処理を行い、コンピュータはシャットダウンされます。

[*23] 字面の通りマウントを解くこと。

第 4 章

Arch Linux を使用する

第 3章で Arch Linux 自体のインストールは完了しました。しかし OS をインストールしたというだけでは明らかに消化不十分で、実用に耐えうるような OS を本ゼミの参加者諸氏は必要としているはずです。Arch Linux のインストールが完了した直後においては、訓練された人間以外はほぼ何も出来ないと言ってもよい、まっさらな環境が眼前に広がるのみです。しかし、このまっさらな環境は、骨格を既に備え、全ての基本となる、頼もしい環境であるともいえます。

本章では Arch Linux を一般的な用途に用いる為の諸工程を扱います。まずインターネットに接続するためのネットワークマネージャの設定、Arch Linux の標準パッケージマネージャである `pacman` の紹介と説明、一般ユーザの追加、各種パッケージのインストールとその設定、またシステムそれ自体の各種設定を扱い、最終的に日本語環境下でのブラウジングを可能とする環境を構築します。また補足として、Arch Linux を使う上で知っておくと便利と思われるものをいくつか紹介しておくことにします。

4.1 netctl の設定

第 3章でも扱った通り、Arch Linux は規定のネットワークマネージャとして netctl を使用します。netctl は CUI ベースのネットワークマネージャで、テキストファイルによる設定ファイルを用いる事で柔軟かつ軽快な動作を可能にしています。また netctl は Arch Linux が抱えるプロジェクトの一つでもあり、このため Arch Linux と netctl の親和性は非常に高いものになっています。

GNU/Linux をはじめとする Unix 系 OS には、他に wicd[*1] や networkmanager[*2] 等のネットワークマネージャがあり、これらは Linux ディストリビューションに依存していない汎用なネットワークマネージャです。もちろん netctl もまた、Arch Linux が抱えるプロジェクトではありますが、汎用のネットワークマネージャです。ここでは Arch Linux との親和性と事前準備に掛かる手間の少なさとから、netctl を使用してネットワークを設定していきます。

netctl は、/etc/netctl 下に配置される設定ファイルを元にネットワークを設定します。何も設定されていない状態では、/etc/netctl は以下のような 3 個のディレクトリが配置されているのみです。ここで “`hoge/`” は hoge という名前のディレクトリを表現していることとします。

```
# cd /etc/netctl && ls
examples/ hooks/ interfaces/
#
```

[*1] `https://launchpad.net/wicd`

[*2] `https://wiki.gnome.org/Projects/NetworkManager`

それでは、設定ファイルを作成しましょう。今回は、現在インターネットをクライアントとして利用する際にはほとんどと言って良い程に広く使われる DHCP を用いた接続を想定しています。本ゼミが扱う範疇には含まれませんが、インターネットを利用する上では DHCP 以外の接続方式も存在します。また IP は IPv4 を利用するものとします。

まず、自分の環境でネットワークアダプタがどのような記号で表されるかを確認します。ネットワークアダプタとは、その名の通りネットワークを利用する際に使用するアダプタで、具体的にはイーサネットとか wifi を利用するためにシステムへ組み込む機械の事を指します。これは以下を叩くことで確認が可能です。

```
# ip a
```

この結果、例えば以下のような結果が得られるでしょう。

```
1:  lo:  <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
valid_lft forever preferred_lft forever
inet6 ::1/128 scope host
valid_lft forever preferred_lft forever
2:  enp0s25:  <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
link/ether 00:26:2d:f5:e6:b0 brd ff:ff:ff:ff:ff:ff
3:  wls1:  <BROADCAST,MULTICAST> mtu 1500 qdisc mq state DOWN group default qlen 1000
link/ether 00:26:c6:c6:32:88 brd ff:ff:ff:ff:ff:ff
```

最近のインストールであれば[*3]、この一覧のうち、“enp” で始まる文字列はイーサネットアダプタ（有線 LAN）を指し、また “wls” で始まる文字列は wifi アダプタ（無線 LAN）を指します。以下では、有線 LAN を使用する場合と無線 LAN を使用する場合とのそれぞれについて説明を行います。ただし、以下の説明ではイーサネットアダプタを enp0s25、wifi アダプタを wls1 と表すこととします。ちなみに、このようなシステム上でデバイスを指し示す文字列をデバイス名と言います。つまり、この例では、イーサネットアダプタのデバイス名は enp0s25、wifi アダプタのデバイス名は wls1 となります。

4.1.1　有線 LAN を使用する場合の設定

設定ファイルのテンプレートは/etc/netctl/examples/下に格納されています。ディレクトリに格納されているテンプレートは複数あり、多岐に渡る接続方法をサポートしていますが、どのテンプレートがどの接続方式を扱うのかは概ねテンプレートファイルの名前を見れば把握できるようになっています。ここでは有線 LAN（イーサネット）で DHCP によりインターネットへ接続するので、以下のように叩きます。

```
# cp /etc/netctl/examples/ethernet-dhcp
/etc/netctl/hoge
```

設定ファイルの名前は一目で判別できるような物を選ぶと後で困らなくて済みます。例えば今回のようにイーサネットアダプタが enp0s25 と表現されるならば、設定ファイルの名前は “enp0s25-dhcp” とするとよいでしょう。

[*3] 厳密に言えば新しめの udev を使用している場合となる。しかし udev は現在 systemd の一部になっている (https://wiki.archlinux.org/index.php/udev) ので、現行の systemd を用いていれば udev も自動的に新しくなる。

設定ファイルを作成したら、次に内容を編集します。適当なエディタで設定ファイルを開くと、図 4.1のような内容になっています。

```
Description='A basic dhcp ethernet connection'
Interface=eth0
Connection=ethernet
IP=dhcp
## for DHCPv6
#IP6=dhcp
## for IPv6 autoconfiguration
#IP6=stateless
```

図 4.1: ethernet-dhcp の内容

今回は IPv4 を使用するので、変更する箇所は “Interface=eth0” の箇所のみです。ここをイーサネットアダプタのデバイス名に置換します。つまり、図 4.2のように内容を編集します。

```
Description='A basic dhcp ethernet connection'
Interface=enp0s25
Connection=ethernet
IP=dhcp
## for DHCPv6
#IP6=dhcp
## for IPv6 autoconfiguration
#IP6=stateless
```

図 4.2: ethernet-dhcp の内容

4.1.2　無線 LAN を使用する場合の設定

無線 LAN を使用する場合も、有線 LAN を使用する場合と基本的な部分は変わりません[*4]。ただし、wifi では通信における種々の暗号化方式がテンプレートとして用意されているので、適切なテンプレートを選択する必要があります。もちろん、テンプレートに頼らず全部自力で設定を行うという方は選択の必要はありませんし、そもそもテンプレートを利用する必要もありません。テンプレートを利用する場合、例えば暗号化方式として WPA を利用する場合は、

```
# cp /etc/netctl/examples/wireless-wpa /etc/netctl/fuga
```

[*4] むしろ無線 LAN を設定する場合、wifi アダプタが適切に認識され、ドライバが読み込まれているか否かによって難易度に劇的な差が出る。前述の `ip a` で wifi アダプタを指す文字列が表示された場合、設定はほぼ終わったものと解釈してよろしい。もし表示されない場合は、`https://wiki.archlinux.org/index.php/Wireless_network_configuration`などを参照してシステムへ wifi アダプタを認識させる必要があるが、この作業は一般に面倒であり、また単一の対処法が存在しない。よって、この詳細はこの文書では割愛する。

のように叩き、内容を編集します。この内容は図 4.3のようになります。

```
Description='A simple WPA encrypted wireless connection'
Interface=wlan0
Connection=wireless
Security=wpa
IP=dhcp
ESSID='MyNetwork'
# Prepend hexadecimal keys with \"
# If your key starts with ", write it as '""<key>"'
# See also:  the section on special quoting rules in netctl.profile(5)
Key='WirelessKey'
# Uncomment this if your ssid is hidden
#Hidden=yes
```

図 4.3: wireless-wpa の内容

この場合も、“Interface=wlan0” の箇所を適切なデバイス名へ変更します。また、“Key='WirelessKey”' の箇所を適切なパスフレーズに変更します。パスフレーズが “Jerusalem” であれば、修正後は図のようになります。

```
Description='A simple WPA encrypted wireless connection'
Interface=wls1
Connection=wireless
Security=wpa
IP=dhcp
ESSID='MyNetwork'
# Prepend hexadecimal keys with \"
# If your key starts with ", write it as '""<key>"'
# See also:  the section on special quoting rules in netctl.profile(5)
Key='Jerusalem'
# Uncomment this if your ssid is hidden
#Hidden=yes
```

図 4.4: wireless-wpa の内容

ただ、以上のように設定こそ可能ですが、無線 LAN を構成する場合、もっと便利な方法があります。

```
# wifi-menu <devname_wifi>
```

と叩くと、インタラクティブな設定画面が表示されます。*devname_wifi* の箇所は wifi アダプタのデバイス名を指定します。

セキュリティ意識の高い方は、平文としてパスフレーズを保管することに嫌忌するかと思います。そんな方のために、パスフレーズを暗号化して保管する方法もあります。具体的な方法は`https://wiki.archlinux.org/index.php/netctl`をご参照ください。

4.1.3　接続

設定ファイルを作成後、以下のように叩くと、自動的にネットワークの構成やネットワークアダプタの設定その他諸々が行われ、インターネットへ接続されます。*config_file* へは設定ファイルの名前を指定します。

```
# netctl start <config_file>
```

この設定を常に利用し、次回起動時も利用する場合は、以下のように叩くとよいでしょう。

```
# netctl enable <config_file>
# systemctl enable netctl
```

また、接続とは直接関係ありませんが、インターネットへの接続設定を容易に他人に見られてしまうのはセキュリティ上好ましくありません。設定を確認できるのはシステムの管理者のみに限るべきです。このようなファイルに関する権限 (permission) の設定を行うには、chmod というコマンドを使用します。今回は以下のように叩くことで、所望の設定を得られます。

```
# chmod go-r <config_file>
```

4.2　pacman

多くの Linux ディストリビューションでは、パッケージの管理のためにパッケージマネージャというプログラムを使用しています。Linux ディストリビューションはディストリビューション毎にパッケージの貯蔵庫とでも言うべきレポジトリというサーバを持っており、パッケージマネージャはこのレポジトリにアクセスし、コマンドとして命令されたパッケージを検索またはインストールしたり、あるいはシステムにインストールされたパッケージをアンインストールするなどの機能を持ちます。

Arch Linux がデフォルトで使用するパッケージマネージャは pacman と呼ばれるものです。本来パッケージマネージャは前述の機能以外に多数の機能を持ち、その例に漏れず pacman も多数の機能を持ちますが、その詳細は割愛し、本文中で pacman を扱う際に、文脈に関係する機能を都度説明していくこととします。

4.2.1　pacman の設定

pacman は設定ファイルとして/etc/pacman.conf を使用します。このファイルは、pacman が使用するレポジトリを指定したり、pacman が利用するオプションを選択するためのものです。

ご多分に漏れずこの設定ファイルもテキストファイルです。今回は GUI での日本語環境構築のためにレポジトリを追加する必要があるため、設定ファイルを編集していきます。

レポジトリは通常図 4.5のような形式で指定されます。

```
[レポジトリ名]
SigLever = <レポジトリに対する認証の程度>
Server = <レポジトリの URL>
```

図 4.5: レポジトリの形式

ただし、Arch Linux 公式レポジトリ (core、community、community-testing、exta、multilib、multilib-testing など) は、“Server = <レポジトリの URL>” の代わりに、“Include = /etc/pacman.d/mirrorlist” を指定します。

今回追加するレポジトリは、GUI 上で日本語を入力するための IME である Mozc[*5] を公開しているレポジトリです。https://wiki.archlinux.org/index.php/Mozcに記載の通り、

```
[pnsft-pur]
SigLevel = Optional TrustAll
Server = http://downloads.sourceforge.net/project/pnsft-aur/pur/$arch
```

を/etc/pacman.conf に追記します。場所はどこでも良いのですが、ファイルの末尾に追記するのがベターです。

4.2.2　pacman を使う

レポジトリを追記したら、以下を叩きます。

```
# pacman -Syu
```

これは Arch Linux を使用する上で頻繁に唱えていくコマンドの一つです。意味としては/etc/pacman.d/mirrorlist で設定された Arch Linux のサーバと同期し、レポジトリの更新、パッケージのアップデートの確認を行うものです。

続いて、sudo というパッケージをインストールします。これは次節で一般ユーザのアカウントを追加した場合、システムに変更を及ぼすような操作が一般ユーザではできないので、一時的に一般ユーザにスーパーユーザの権限を与えるためのプログラムです。

インストールには、以下のように叩きます。

```
# pacman -S sudo
```

さて、パッケージのインストールを扱ったからには、パッケージのアンインストールの方法も扱っておかないと均衡が取れないような気がするので、アンインストールのためのコマンドも併せて扱っておきましょう。

```
# pacman -Rs PACKAGE_NAME
```

と叩くことで、*PACKAGE_NAME* をアンインストールすることができます。PACKAGE_NAME に関連する全てのパッケージ、データなどを削除したい場合は、

```
# pacman -Rsc PACKAGE_NAME
```

とすると良いでしょう。

4.3　sudo の設定

さて、sudo を一般ユーザに有効化する場合、一般にユーザを wheel というグループに追加する必要があります。グループの追加に関しては次節で扱いますが、ここでは sudo の設定ファイルを編集する visudo を用いて、予め wheel グループに対して sudo を有効化しておくことにしましょう。

visudo とは/etc/sudoers を編集するファイルですが、visudo をそのまま叩くと vi というエディタが起動し、慣れ

[*5] https://code.google.com/p/mozc/

ない人が使うと訳が分からずパニックに陥る可能性があります。ここではエディタとして nano を指定し、安心して編集することができるようにします。このためには

```
# EDITOR=nano visudo
```

と叩きます。

さて、どこを編集するかですが、ファイル中にある

```
## Uncomment to allow members of group wheel to execute any
command
#%wheel ALL=(ALL) ALL
```

という記述の 2 行目、即ち “#%wheel ALL=(ALL) ALL” をアンコメントします。これにより、コメント文にもあるとおり、wheel グループに所属するユーザに sudo が有効となります。

visudo による変更はシステム再起動後でないと有効になりません。ですが、ユーザをグループに追加するという操作もシステムの再起動後に有効になるので、ついでにユーザアカウントについての操作を行ってから再起動することにしましょう。

4.4　ユーザアカウント

4.4.1　一般ユーザアカウントの追加

現在システム上にはスーパーユーザしか存在せず、これは非常にまずいことであることは第 3章で説明しました。ここで一般ユーザのアカウントをシステムに追加し、以後の作業は一般ユーザのアカウントを用いて行うことにしましょう。

ユーザアカウントを追加するには、以下を叩きます。

```
# useradd -m -g users -s /bin/bash USERNAME
```

オプションの意味はhttps://wiki.archlinux.org/index.php/Users_and_Groupsをご参照ください。これにより、一般ユーザアカウントの *USERNAME* が作成され、/home 下に/home/*USERNAME* というディレクトリが作成されます。この/home/*USERNAME* というディレクトリが、*USERNAME* というユーザのホームディレクトリ[*6] となります。

アカウントを作成したら、次にアカウントのパスワードを設定します。

```
# passwd USERNAME
```

と叩き、*USERNAME* に対するパスワードを決定します。パスワードの設定手順は第 3章で root アカウントのパスワードを設定した際と同様になります。

パスワードを設定したら、危険極まるスーパーユーザのアカウントから離脱しましょう。exit を用いてシステムからログアウトし、新たに作成した一般ユーザのアカウントで再ログインします。

ここで、シェルの表示が

```
[USERNAME@HOSTNAME ~]$
```

[*6] デスクトップと同義。

となっていることに注目しましょう。この表示のうち、“~” は今自分がユーザのホームディレクトリ、即ち /home/*USERNAME* にいることを意味しています。また、シェルのプロンプトが “#” から “$” に変化しています。これは、現在システムへログインしているユーザがスーパーユーザ (#で表される）から一般ユーザ（$で表される）に変わったためです。ここで、試しに指定したディレクトリへ移動する cd を用いて “/” へ移動してみましょう。すると

```
[USERNAME@HOSTNAME /]$
```

というように表示が変化するはずです。このように、シェル上では今自分がどのディレクトリにいるかがすぐにわかるようになっています。ちなみにホームディレクトリへ戻るには、

```
$ cd ~
```

または

```
$ cd
```

と叩きます。

以下では、スーパーユーザの権限（管理者権限ともいう）が必要な操作の場合はプロンプトを “#”、それ以外の場合はプロンプトを “$” とすることで表現します。

4.4.2　ユーザをグループに追加する

前節で扱ったとおり、追加した一般ユーザに sudo を有効化するためには、ユーザを wheel グループへ追加することが必要です。加えて、GUI を使用する等グラフィック関連のデバイスを操作するためにはユーザを video グループへ追加する必要がありますし、スピーカから音を鳴らす等の音声関連のデバイスを操作するためにもユーザを audio グループへ追加する必要があります。グループというのは、特定の権限を許可された集合であると解釈するのが適当でしょう。

では早速ユーザをグループへ追加していきます。これには gpasswd というコマンドを用います。ユーザをグループへ追加するという行為はシステム全体へ影響を及ぼすため、一般ユーザでの行使は許可されません。現状は sudo を利用できない一般ユーザという身分ですので、

```
$ su
```

と叩くことでスーパーユーザへ移行することにします。ここで要求されるパスワードはスーパーユーザのパスワードです。

ユーザをグループへ追加するには以下のように叩きます。

```
# gpasswd -a USERNAME GROUP_NAME
```

これは *USERNAME* というユーザを *GROUP_NAME* という名前のグループへ追加 (“-a”) することを意味しています。このようにして、wheel、video、audio グループへユーザを追加しましょう。

ユーザのグループへの追加は再ログイン後に有効になります。ここまでの操作を完了したあとは

```
# exit
```

と叩き、ログアウト後、再び一般ユーザとしてログインしましょう。

4.5　GUI の導入

今後は一般ユーザでシステムにログインしていることを前提にして説明を進めます。具体的にはスーパーユーザ権限を必要とするコマンドには sudo をコマンドの先頭に用いることで示します。これが sudo の使用方法です。

さて、GUI には通常 X window system（以下 X）というソフトウェアを使います。X は GUI を提供するソフトウェアです[*7]。以後、X のインストールによって GUI の基礎を構築し、X 上で動作するメジャーなデスクトップ環境のひとつである Xfce をインストールすることで、馴染み深い GUI 環境を構築することを目指します。

4.5.1　X の導入

Arch Linux で X を動作させるには、最低限以下のコマンドを叩いてインストールされるパッケージが必要です。

```
$ sudo pacman -S xorg-server xorg-server-utils
xorg-xinit
```

X の動作確認には、加えて以下で得られるパッケージが必要になります。

```
$ sudo pacman -S xorg-twm xterm xorg-xclock
```

続いてグラフィックスドライバをインストールします。GUI を使用するにはコンピュータのグラフィックス機能を利用します。そのためには当然グラフィックドライバが必要になります。VirtualBox Extension Pack をインストールしてあれば、vesa というオープンソースの汎用グラフィックドライバを導入することで対応できます。Arch Linux では以下のように叩いてインストールします。

```
$ sudo pacman -S xf86-video-vesa
```

これで X を動作させる環境が一応整ったはずです。

```
$ startx
```

と叩き、図 4.6のような出力になるかを確認します。

4.5.2　GUI におけるキーマップの再設定

第 3章でも触れた通り、CUI で設定したキー配列は GUI では有効にならないことがあります。GUI においても適切なキー配列を設定する必要があるでしょう。ここではシステム再起動まで有効である一時的な設定と、永続的に有効である設定との 2 つを紹介します。

一時的な設定

とりあえず一時的でいいからキー配列を変えたいという場合、setxkbmap というコマンドを用いることができます。例えはキー配列を JIS 配列にしたい場合、引数に “jp” を指定して、

[*7] 詳しく言えば X はソフトウェアの名称ではなく通信プロトコルの名称である。GUI は X という通信プロトコルを用いて提供される、と表記するのが正しい（詳細はhttp://ja.wikipedia.org/wiki/X_Window_Systemを参照）。

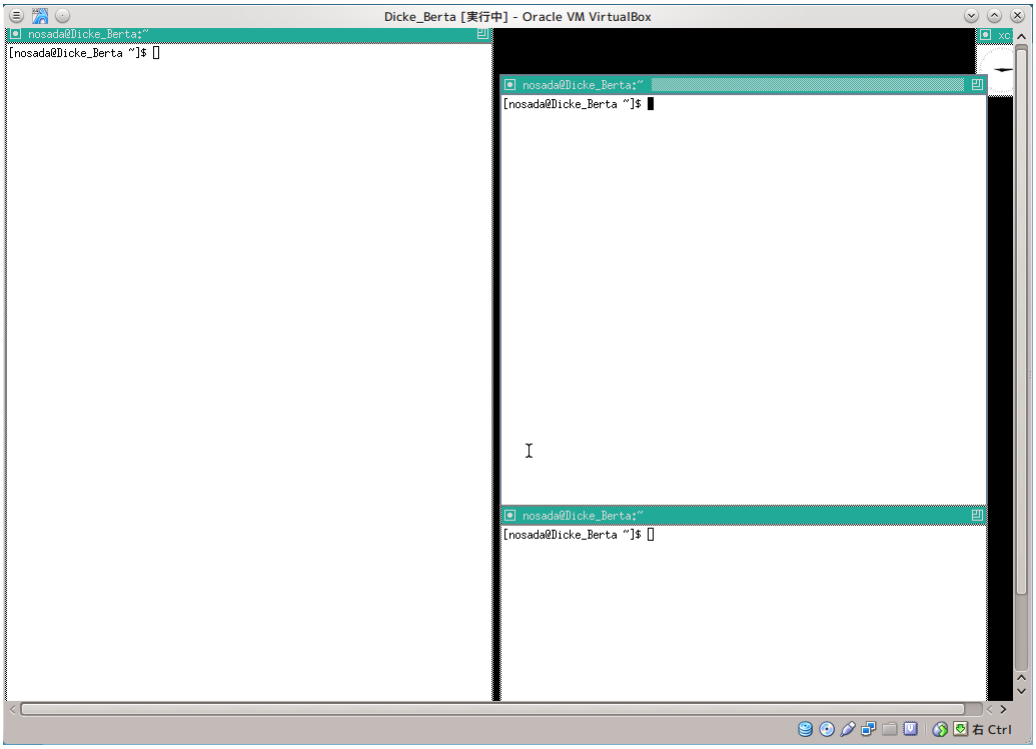

図 4.6: startx

```
$ setxkbmap jp
```

とすると良いでしょう。

永続的な設定

キー配列の変更を明示的に変更するには、X の設定ファイルを編集する必要があります。X の設定ファイルの場所は、概ね/etc/X11/xorg.conf.d です。ここを `ls` で確認すると、例えば 10-evdev.conf や 10-quirks.conf といった名前のファイルがあると思います。ここに置かれる設定ファイルは、名前が “<数値>-<名前>.conf” という命名規則に従うことを要求されます。このうち名前は適当でいいのですが、数値に関しては、既存のファイルに振られた数値よりさほど差がない値が良いでしょう。上の例であれば例えば 20 や 30 などが適当です。

さて、ここでは設定ファイルを “20-jpkbdmap.conf” として作成することにしましょう。名前は明快に “JaPanese KeyBoarD MAPping” です。ここに、

```
Section "InputClass"
        Identifier "Keyboard Deults"
        MatchIsKeyboard "yes"
        Option "XkbLayout" "jp"
        Option "XkbModel" "jp106"
EndSection
```

と記述します。GUI を再起動すれば、設定は有効になるはずです。

4.5.3　Xfce のインストール

Xfce[*8] とは、前述したようにデスクトップ環境と呼ばれる、GUI を提供するソフトウェアです。X も GUI を提供するソフトウェアだと説明しましたが、例えるなら X はコンクリート打ちっぱなしの内壁で、デスクトップ環境は断熱材や遮音材などを挟んで内装を整え、壁紙を貼った内壁のようなものです。すなわち X は GUI の基礎を構築し、X の構築した GUI を用いて、デスクトップ環境は GUI を提供するのです。

Xfce をインストールするには、以下のように叩きます。

```
$ sudo pacman -S xfce4 xfce4-goodies
```

4.5.4　LightDM のインストール

LightDM[*9] とは、軽量なログインマネージャです。ログインマネージャとはシステムへログインするインターフェースの GUI 版で、機能は CUI のログインインターフェースとそれほど変わるものではありません。ですが、ログインマネージャを導入した場合、マシンの電源を投入するとデスクトップ環境の起動に至るまでにコマンドを叩いたりする操作が不要になるので、使い勝手が少々よくなるという利点があります。

LightDM のインストールには、以下を叩きます。

```
$ sudo pacman -S lightdm lightdm-gtk3-greeter
xorg-server-xephyr
```

インストール後、

```
$ sudo systemctl enable lightdm.service
```

と叩き、システム起動時に LightDM が起動するよう設定します。設定後、システムを再起動させれば、少々みすぼらしいですが、GUI 上でログインが可能になっているはずです。

4.5.5　Xfce の設定

では早速 Xfce を設定していきましょう。今回は本ゼミの目標である「ブラウザで google を開いて日本語で検索ワードを入力して検索する」を達成する上で触れなければならない事柄について扱うにとどめます。他に必要があれば、本章末尾の補足で扱いたいと思います。

ちなみに、Xfce で CUI によるコマンド操作を行うには、Xfce 上でシェルを使用する必要が有ります。Alt+F2 を入力してランチャを起動し、

```
xfce4-terminal
```

とすることで、馴染み深い CUI が立ち上がります。ここでプログラムの起動に使用したのはシェルではなくランチャでありますので、上の操作においてプロンプトが表示されない事に注意して下さい。

[*8] http://www.xfce.org/

[*9] http://www.freedesktop.org/wiki/Software/LightDM/

ロケールの再設定

第 3章で、CUI でのロケールは英語、GUI のロケールは日本語と個別に設定する、と述べましたが、どうも Xfce ではこのような設定は不可能なようです[*10]。予想外の事態なのですが、思い悩んでも埒が明かないので、おとなしく/etc/locale.conf を再設定しましょう。内容を “LANG=”ja_JP.UTF-8”” に修正し、再起動します。ちなみに単純に

```
$ nano /etc/locale.conf
```

とすると、/etc/locale.conf を編集できません。というのも、Linux 上の全てのファイルにはファイルの管理者とパーミッションが定められており、基本的にシステムの設定ファイルはスーパーユーザが管理者となっています。加えて設定ファイルのパーミッションは、スーパーユーザに対してのみ書き込みが可能となっているので、一般ユーザが設定ファイルを編集する場合は sudo を用いなければなりません。

再起動後、システムは見事に日本語化されていると思います。

Uim と Mozc の導入

次に日本語を入力するための IM[*11] である Uim[*12] と、おそらく現在オープンソースの IME[*13] の中で最も性能の良いと思われる Mozc[*14] をインストールします。

これには以下を叩きます。

```
$ sudo pacman -S uim uim-mozc mozc
```

インストール後、Uim で Mozc を使用するための設定を行います。まず CUI 上で

```
$ sudo uim-module-manager --register mozc
```

と叩き、Uim に Mozc を登録します。続いて

```
$ uim-pref-gtk
```

を叩き、UIM の設定ウィンドウを呼び出します。次に、「全体設定」中の「入力方式の利用準備」内にある「標準の入力方式を指定する」にチェックを入れ、「標準の入力方式」のプルダウンメニューから Mozc を選択します。

続いて、Uim がシステム起動時[*15] に自動的に立ち上がるように設定します。/home/*USERNAME*/.xprofile（~/.xprofile と同義）の末尾に以下を追記します。

```
export GTK_IM_MODULE='uim'
export QT_IM_MODULE='uim'
uim-xim &
export XMODIFIERS='@im=uim'
uim-toolbar-gtk-systray &
```

[*10] 筆者が普段使用している KDE では可能。理由がわからない。
[*11] Input Method。文字通り多言語を「入力する手段」である。
[*12] https://code.google.com/p/uim/
[*13] Input Method Engine。言語を入力するために IM 中で駆動させるプログラム。
[*14] https://code.google.com/p/mozc/
[*15] 正確にはデスクトップ環境の起動時。

ただし、初期状態では恐らく当該ファイルはディレクトリ上に存在しないので、

```
$ nano ~/.xprofile
```

として、.xprofile を作成する必要があります。この方法でファイルを作成すると実際にファイルへの変更を保存するまでファイル自体は作成されないことに注意して下さい。なお、.xprofile のようにファイルの先頭に “.”（ドット）がつくファイルは隠しファイルと呼ばれるもので、通常の手段（単に `ls` と叩くなど）では表示されません。隠しファイルを表示させる場合、`ls` では`-a` オプションを付けます。GUI 上のファイラではファイラの操作方法に依存するので一概に言えません。

追記後、システムを再起動させれば、日本語入力が可能となっているはずです。日本語入力の入切は全角/半角キーで、Uim 自体の入切は Alt+Space で可能です。キーバインドは任意で変更可能ですから、設定を使いやすいように変更するのも良いでしょう。

ブラウザの導入

最後にブラウザの導入です。ブラウザには Midori[*16] を使用します。Midori は

```
$ sudo pacman -S midori
```

とすることでインストールができます。

ブラウザの起動方法は、デスクトップ右上の「アプリケーションメニュー」から「インターネット」を選択し起動するのも良いですが、`xfce4-terminal` を起動したように Alf+F2 を入力した後

```
midori
```

と叩いても起動することができます。

では、いよいよ本ゼミの最終目標である、「ブラウザで google を開いて日本語で検索ワードを入力して検索する」を実践したいと思います。とはいっても普段皆さんが普段使用しているブラウザで行っているのと同様に、ブラウザのアドレスバーに`http://www.google.com`と入力し、表示される google の検索ボックスに適当な検索ワードを入力し、Enter を押下するだけです。これで、Arch Linux 上に構築した GUI においてブラウザを起動し、ブラウザで google にアクセスし、google の検索ボックスに日本語で検索ワードを入力し、検索を行う、という本ゼミの目標を達成することが出来ました。

4.6　補足

ここでは Arch Linux を常用する上で知っておくべきであろう事柄について、簡単に触れていきたいと思います。

この節では説明を簡略化する代わりに、各項目に対応した ArchWiki の URL を添えておきます。興味をお持ちになった際は、ArchWiki の当該項目で詳細をご覧ください。

[*16] `http://www.midori-browser.org/`

図 4.7: やっと終わった。

4.6.1　AUR の利用 (https://wiki.archlinux.org/index.php/AUR)

Arch Linux には、Arch Linux 本家が管理運用するレポジトリの他に、Arch Linux のユーザがパッケージを作成するためのファイル (PKGBUILD) を投稿する AUR(Arch User Repository)[*17] というものがあります。ここには主にライセンスの都合上本家のレポジトリには登録できないパッケージや、本家に登録されているパッケージの不安定版、あるいは既存のパッケージに機能を追加するためのパッケージであったり、ジョークプログラムを収めたパッケージであったりと、多種多様なパッケージが登録されています。この多種多様というワードが示す通り、使用したいと思ったパッケージが公式レポジトリになくても、AUR を探せば大抵見つかるという便利さも兼ね揃えています。

この AUR ですが、残念ながら pacman から直接では利用することができません。利用する場合、AUR にアクセスし、インストールしたいパッケージを探し、そのパッケージを作成するための PKGBUILD が含まれた圧縮ファイルを入手し、解凍、パッケージ作成、インストールという手順を踏まなければなりません。

ですが、pacaur[*18] という pacman のラッパプログラムを用いれば、pacman と同じような使い方で AUR にアクセス、PKGBUILD をダウンロードすることができます。

ところで、おそらく図 4.7はあなたが構築したデスクトップ環境に比べ、フォントが綺麗なものになっていると思います。AUR にはこのようなフォントデータも大量に転がっており、わざわざインターネット上からフォントを探し出してインストールしなくても、シェル上でコマンドを叩くだけで導入が可能なのです。

[*17] https://aur.archlinux.org/
[*18] https://wiki.archlinux.org/index.php/Pacaur

4.6.2　ネットワーク接続の管理 (https://wiki.archlinux.org/index.php/Network_Configuration)

実機への Arch Linux を行った場合、特にノートパソコンなどではインターネットへの接続に無線接続を用いることが多いと思います。そのような場合などの汎用性を鑑みて、補足として wicd[*19] というネットワーク接続管理ソフトウェアのインストールを扱います。この操作は必ずしも必須ではありません。

まず以下のように叩き、wicd と wicd の動作に必要なパッケージをインストールします

```
$ pacman -S wicd
```

インストールが終了したら、次に以下のように叩き、wicd デーモンを有効化します。

```
# systemctl enable wicd.service
```

最後に、wicd 以外のネットワークデーモンを停止、無効化します。これは wicd 以外のネットワークデーモンが動作している場合、wicd の動作に支障を来すおそれがあるためです。この操作と上の操作は、環境によって前後する場合があります。適宜調整してください。具体的にはこの文書の順序に従ってエラーが発生した場合、順番を入れ替えて実行してください。

```
# systemctl stop dhcpcd.service
# systemctl stop netctl.service
# systemctl disable dhcpcd.service
# systemctl disable netctl.service
```

これで使用準備は完了です。実際に使用可能になるのはインストール完了後に再起動した後になります。

wicd を CUI で使用する場合は

```
$ wicd-curses
```

と叩くとインターフェースが起動します。GUI で使用する場合は、例えば今回のように Xfce などの GTK 系のツールキットを使用しているデスクトップ環境の場合は、

```
$ sudo pacman -S wicd-gtk
```

と叩き、`wicd-gtk` を使用すると良いでしょう。

無線接続などの詳細な設定は扱いません。`https://wiki.archlinux.org/index.php/Wireless_Setup`を参照するのが手っ取り早いでしょう。wicd については`https://wiki.archlinux.org/index.php/Wicd`が役立ちます。

[*19] `http://wicd.sourceforge.net/`

4.6.3　音を鳴らす (https://wiki.archlinux.org/index.php/Sound)

Arch Linux 上でサウンドシステムを扱うには、ALSA[*20] と OSS[*21] の 2 種類の手段があります。通常は Linux カーネルの構成部品の一つである ALSA を使用するのが良いでしょう。ここでは ALSA について触れます。

ALSA を使用するには、

```
$ sudo pacman -S alsa-utils
```

と叩いて ALSA を触るためのユーティリティをインストールし、

```
$ alsamixer
```

を叩きます。CUI で表示されるミキサー画面を適当に弄くり回せば十分でしょう。デフォルトでは全てのパラメータがミュートされていますが、M キーによりミュートをオン・オフできます。

ALSA については https://wiki.archlinux.org/index.php/Advanced_Linux_Sound_Architecture を、OSS に関しては https://wiki.archlinux.org/index.php/Open_Sound_System を参照すると良いでしょう。

[*20] http://www.alsa-project.org/main/index.php/Main_Page
[*21] http://www.opensound.com/

参考文献

[1] Robert Love 著, 千住治郎 訳, 『LINUX システムプログラミング』(オライリー・ジャパン, 2011)

[2] Abraham Silberschatz, Peter Baer Galvin, Greg Gagne,"Operating System Concepts 9th Edition"(John Wiley & Sons, Inc., 2013)

[3] 岡田健治, 川井義治, 宮原徹, 佐久間伸夫, 遠山洋平, 田口貴久, 松田神一, 木村真之介, 高橋征義, 『Linux 標準教科書 (ver2.0.0)』(LPI–Japan, 2012)
http://www.lpi.or.jp/linuxtext/text.shtml

[4] 『Beginners' Guide - ArchWiki』
https://wiki.archlinux.org/index.php/Beginners'_Guide

[5] 『Installation Guide - ArchWiki』
https://wiki.archlinux.org/index.php/Installation_Guide

[6] 『Arch Linux - ArchWiki』
https://wiki.archlinux.org/index.php/Arch_Linux

[7] 『Linux - Wikipedia』
http://ja.wikipedia.org/wiki/Linux

[8] 『UNIX - Wikipedia』
http://ja.wikipedia.org/wiki/UNIX

[9] 『History of Linux - Wikipedia, the free encyclopedia』
http://en.wikipedia.org/wiki/History_of_Linux

[10] 『About Linux Kernel』
https://www.kernel.org/linux.html

[11] 『GNU - Wikipedia』
http://ja.wikipedia.org/wiki/GNU

[12] 『GNU オペレーティング・システムについて - GNU プロジェクト - フリーソフトウェアファウンデーション』
http://www.gnu.org/gnu/about-gnu.html

[13] 『自由ソフトウェアとは? - GNU プロジェクト - フリーソフトウェアファウンデーション』
http://www.gnu.org/philosophy/free-sw.html

[14] 『BSD - Wikipedia』
http://ja.wikipedia.org/wiki/BSD

[15] 『Minix - Wikipedia』
http://ja.wikipedia.org/wiki/Minix

[16] 『Linux distribution - Wikipedia, the free encyclopedia』
https://en.wikipedia.org/wiki/Linux_distribution

[17] 『Filesystem Hierarchy Standard』

http://www.pathname.com/fhs/pub/fhs-2.3.html

[18]『Arch filesystem hierarchy - ArchWiki』
https://wiki.archlinux.org/index.php/Arch_filesystem_hierarchy

[18]『Partitioning - ArchWiki』
https://wiki.archlinux.org/index.php/partitioning

[19]『netctl - ArchWiki』
https://wiki.archlinux.org/index.php/netctl