

Windows で Unix

2021 年 12 月 8 日

目次

1	このドキュメントについて	6
1.1	はじめに	6
1.2	連絡方法	6
1.3	利用条件	6
1.4	注意事項および免責事項	6
2	英語学習	6
2.1	はじめに	6
2.2	英語学習の手順	7
2.3	英語の発音・リスニング	7
2.4	英語の文法・語彙・会話表現	10
2.5	英語発音の注意点	12
3	情報セキュリティ	14
3.1	はじめに	14
3.2	システムのセキュリティ	14
3.3	通信のセキュリティ	15
3.4	メディアのセキュリティ	15
4	フォントとアイコン	16

4.1	はじめに	16
4.2	フォント	16
4.3	アイコン	17
5	ファイルとアプリケーション	17
5.1	はじめに	17
5.2	テキストファイルの種類	17
5.3	アーカイブファイルのフォーマット	17
5.4	イメージファイルのフォーマット	18
5.5	オーディオファイルのフォーマット	20
5.6	ビデオファイルのフォーマット	21
5.7	コンテナファイルのフォーマット	23
5.8	テキストファイルを扱うアプリケーション	24
5.9	アーカイブファイルを扱うアプリケーション	24
5.10	イメージファイルを扱うアプリケーション	24
5.11	オーディオファイルを扱うアプリケーション	24
5.12	ビデオファイルを扱うアプリケーション	25
6	ワイルドカードと正規表現	25
6.1	はじめに	25
6.2	ワイルドカード	25
6.3	正規表現	26
7	Windows のカスタマイズ	29
7.1	はじめに	29
7.2	パーティションの分割	29
7.3	パーティションの暗号化	30
7.4	システムのカスタマイズ	30
7.5	タスクバーのカスタマイズ	31
7.6	デスクトップのカスタマイズ	31
7.7	キーバインドのカスタマイズ	31

7.8	アプリケーションのインストール	31
7.9	ウェブブラウザのカスタマイズ	32
8	かな入力のカスタマイズ	33
8.1	はじめに	33
8.2	月配列の特長	33
8.3	月配列の種類	33
8.4	月配列の導入	33
9	DSP のカスタマイズ	34
9.1	はじめに	34
9.2	DSP の機能	34
9.3	DSP の使用例	36
10	ファイルのバックアップ	37
10.1	はじめに	37
10.2	バックアップの準備	37
10.3	バックアップの実行	38
11	レジストリのバックアップ	39
11.1	はじめに	39
11.2	レジストリ全体のバックアップ	39
11.3	レジストリキーのバックアップ	39
12	T_EX と文書作成	40
12.1	はじめに	40
12.2	T _E X の利点	41
12.3	T _E X の導入	41
13	ローカルウィキ	41
13.1	はじめに	41
13.2	サーバとクライアント	42

13.3	Hugo	42
13.4	XAMPP/MediaWiki	42
14	オーディオファイルの取り扱い	45
14.1	はじめに	45
14.2	オーディオファイルのノーマライズ	45
14.3	オーディオファイルのタギング	46
14.4	オーディオファイルのリネーム	48
14.5	オーディオファイルのエンコード	48
14.6	オーディオ CD のリッピング	51
15	Unix と Windows の違い	51
15.1	はじめに	51
15.2	ディストリビューション	52
15.3	ユーザインターフェース	52
15.4	ユーザとグループ	52
15.5	ランレベルとデーモン	52
15.6	ブートローダ	52
15.7	ファイルシステム	53
15.8	ディレクトリ構造	54
15.9	ホームディレクトリ	54
15.10	設定ファイル	54
15.11	アーカイブファイル	54
15.12	デバイスファイル	55
15.13	パッケージシステム	55
15.14	ウィンドウシステム	55
15.15	日本語入力システム	55
15.16	サウンドシステム	55
15.17	印刷システム	56
15.18	デスクトップ環境	56

16	Unix の基本操作	56
16.1	はじめに	56
16.2	MSYS2 のインストール	56
16.3	ターミナル (mintty)	57
16.4	パッケージマネージャ (pacman)	60
16.5	コマンドシェル (bash)	61
16.6	テキストエディタ (vim)	64
16.7	メールクライアント (mutt)	70
16.8	ウェブブラウザ (w3m)	71
16.9	ダウンロードマネージャ (curl, wget, aria2)	73
16.10	FTP クライアント (lftp)	73
16.11	JavaScript シェル (V8)	75
16.12	電卓 (bc)	76
17	Linux のインストール	76
17.1	はじめに	76
17.2	BIOS/MBR と UEFI/GPT	77
17.3	デバイスマッパーと LUKS/LVM	77
17.4	Arch Linux のインストール	78
18	コーディングスタイル	89
18.1	はじめに	89
18.2	コメント	89
18.3	空白文字	89
18.4	識別子	89
18.5	制御文	91
18.6	定義文	92
18.7	コードフォーマッタ (ClangFormat)	93
19	Windows プログラミング	94
19.1	はじめに	94

19.2 プログラミングの予備知識	94
19.3 アプリケーションの開発環境	96
19.4 プログラミングの作業工程	97
19.5 プログラミングの学習方法	109

1 このドキュメントについて

1.1 はじめに

ウェブに散らばっている断片的な情報を自分のために分かりやすくまとめているドキュメントです。記事の内容は加筆や修正によって変わることがよくありますがあまり気にしないでください。

1.2 連絡方法

感想や意見はメール (foo02@outlook.jp) で受け付けていますが返信は期待しないでください。

1.3 利用条件

コンテンツは[クリエイティブ・コモンズ 表示-非営利-継承](#)に従ってご利用ください。

1.4 注意事項および免責事項

自分の知識の更新と記事の更新がリンクしているので間違いをそのまま掲載している可能性があります。間違いが見つければすぐに訂正するようにしていますが、記事の内容を鵜呑みにはしないでください。間違った記事の内容によって損害や不利益が生じても筆者は責任を負いません。

2 英語学習

2.1 はじめに

英語をマスターするには訓練を重ねることによってネイティブに近い発音と言語感覚を身につける必要があります。しかし、日本語と英語の間には言語的に大きな差異があり、他の国の人々と比べて日本人は英語の習得に多くの時間 (およそ 2400 時間) を必要とします。また、日

本では英語に触れる機会が少なく英語に親しむためには意識的に英語に接する必要があります。ここでは英語を学ぶにあたって何をすべきかを説明します。語学に王道はありませんが正しい学び方はあります。

2.2 英語学習の手順

英語に限らず語学において最初に取り組むべきことは発音です。英語には日本語にはない発音があります。それらの発音をマスターしなければ、英語を話すことはおろか聞き取ることもできません。英文もスムーズに読めないの読解力も低下します。

ではどの段階で発音をマスターしたと判断すべきでしょうか。試しに [YouTube](#) で英語が話されている動画を再生してください。英語の字幕を追いながら英語の音を聴いてみて両者が一致していると判断できれば問題ないといえるでしょう。

いくら発音が大事だからといっても文法と語彙をおろそかにしてはいけません。正しい文法と豊富な語彙を身につけることによって英語を正しく解釈することができます。学習の順番としては発音の習得が先ですが、発音・文法・語彙のそれぞれを切り離して考えることはできません。

2.3 英語の発音・リスニング

英語と日本語の発声方法は全く異なります。大きな違いは次の2つです。

英語	日本語
腹式呼吸	胸式呼吸
喉発音	口発音

英語のネイティブは腹式呼吸で発声するので声がよく響きますが、多くの日本人は胸式呼吸で発声するので口ごもっているように感じられます。加えてネイティブは喉から音を出しますが日本人は口の先から音を出します。日本人の話す英語がネイティブのように聞こえないのはこれらの違いによるためです。

腹筋に少し力を入れて声を出せば腹式呼吸で発声できます。腹式呼吸で発声すると口の中で音がよく響き、息を強く押し出すことができます。慣れてくれば意識しなくても自然に腹式呼吸で発声できるようになります。

まずは英語の一つ一つの音(音素)が出せるように練習してください。母音は喉を鳴らすようにして音を出します。子音の摩擦音と破裂音はその音が自分の耳ではっきり聞こえるくらい強

く息を押し出します。発音記号で表される英語の音 (音素) とその発音方法について以下に示します。説明の便宜上英語の音をカタカナで表現していますが、本来の英語の音とは区別するようにしてください。

発音記号	発音方法	備考
母音	母音は喉から音を出す (アクセントのある母音は少し延ばす)	
e	エと発音する	
i	イと発音する	
u	ウと発音する	
æ	口角を少し上げる感じでアと発音する	エとアの間の音と言われるがエの音に近い
ɑ	オの口の形でアと発音する	アとオの間の音に聞こえる
ɔ	オと発音する	
ə	軽く喉を鳴らす	あいまい母音 (schwa) といわれる
ɪ	口に力を入れずにイと発音する	イとエの間の音に聞こえる
ʊ	口に力を入れずにウと発音する	ウとオの間の音に聞こえる
ʌ	アと発音する	
子音	子音はカタカナの母音が鳴らないように注意する	
b	ブと発音する	
d	ドと発音する	
dʒ	ヂと発音する	
f	上の前歯の先を下唇につけて息を押し出す	
h	フと発音する	

j	下あごを少し前に突き出しイと発音する
k	クと発音する
l	母音の前では舌尖を上の前歯の裏と上あごの間あたりに押し当てて喉を鳴らす (子音の前や末尾では口に力を入れずにウと発音する)
m	ムと発音する
n	ヌと発音する
p	プと発音する
r	舌を少し巻いて喉を鳴らす (舌が上あごにつかないように注意する)
s	スと発音する
t	トと発音する
tʃ	チと発音する
v	上の前歯の先を下唇につけて息を押し出しながら喉を鳴らす
w	口をすぼめてウと発音する
z	ズと発音する
ð	舌尖を上の前歯の先につけて息を押し出しながら喉を鳴らす (舌尖を上の前歯の裏で弾いて d に近い音を出すこともある)
ŋ	鼻にかけてンと発音する (ハミングのように鼻を鳴らす)
g	グと発音する
ʃ	シュと発音する
ʒ	ジと発音する
θ	舌尖を上の前歯の先につけて息を押し出す (舌尖を上の前歯の裏で弾いて t に近い音を出すこともある)

次に単語の発音を練習します。単語の発音で大切なのはアクセントです。アクセントのある母

音は少し延ばすように発音してください。また発音記号だけでなく実際の発音も確認しましょう。実際の音を聞いてそれを真似るようにします。

最後に文章の発音を練習します。英語を話すときは口にあまり力を入れず少し半開きの状態にします。舌は舌先が口の中央に位置するように少し上げておきます。発音のしやすさを考えれば自然とこのような口の状態になるはずです。文章の発音では音のつながり(リンキング)に注意してください。また、実際の会話では速くスムーズに発音するために音が消えたり変化したり(リダクションおよびフラッピング)します。

リンキング	単語の末尾の子音と単語の先頭の母音がつながって発音される	an apple はアナッポウのように発音される
リダクション	調音点(音を出す位置)が同じ音が続くと前の音が無音もしくは促音のようになる	get to はゲットゥのように発音される
	his/him/her/have/had の h は発音されないことがある	like her はライカのように発音される
	of の f は発音されないことがある	out of はアウトのように発音される
フラッピング	t の音が d もしくはラ行に近い音になる	get it はゲディッもしくはゲリッのように発音される

上記の詳細については[英語で悩むあなたのために](#)を参照してください。

発音方法をマスターしたら [VOA Learning English](#)(アメリカ英語)/[BBC Learning English](#)(イギリス英語) などの動画を見ながらひたすら英語の音のインプットとアウトプットを繰り返します。ネイティブの話し方を真似て同じように話せるようになってください。なお、英語の音のリズムは日本語と違い裏拍子なので音の出だしがワンテンポずれます。この違いを意識すると英語の音が聞き取りやすくなります。

2.4 英語の文法・語彙・会話表現

文法を学ぶ上で大切なことは、英語における物の見方や考え方を理解することです。英語では結論を先に表明するため主語の次に述語が来ます。また、対象が数えられるのか、数えられるならいくつあるのかを常に意識します。このような言語感覚には多くの英語表現に触れることで慣れるしかありません。文法は細かい理屈よりもその感覚を身につけることのほうが大切

です。

助動詞と前置詞にはたくさんの意味や用法がありますが大本の意味をイメージで捉えるようにしてください。例えば辞書で as という単語を引くと多くの意味が掲載されていますが、「A as B とは “A = B” である」ということさえ知っていれば as の持つ多くの意味・用法を容易に捉えることができます。次のような似ている英語表現のニュアンスの違いも分かるようになり、正確な英語表現ができるようになります。

I will return. (私は戻ってくるだろう)	will は個人の意思を表す
-----------------------------	----------------

I shall return. (私は戻ってくる運命にある)	shall は神の意思を表す
--------------------------------	----------------

I will(can) return. (私は戻ってくる)	現在形は主観的なニュアンスになる
-------------------------------	------------------

I would(could) return. (私は戻ってくるだろう)	過去形は客観的なニュアンスになる
-------------------------------------	------------------

法助動詞 (will/can/shall/may/must) の法とは話し手の一時的な感情や精神状態を意味します。過去形 (would/could/should/might) であっても現在や未来のことを表現するために使用されることがほとんどです。過去形の法助動詞で過去のことを表現するには「法助動詞 + have + 過去分詞」という形にします。過去形の法助動詞の意味・用法は推量と仮定法に大別されますが、形が同じなのでどちらの意味になるのかは文脈から判断することになります。

He should have gone there.	推量として解釈すると「彼はそこへ行ったはずだ」という意味になる
----------------------------	---------------------------------

He should have gone there.	仮定法として解釈すると「彼はそこへ行くべきだった」という意味になる
----------------------------	-----------------------------------

文法の詳細については[英文法まとめ！イラストでわかる英語文法の全知識/会話に使える！英文法](#)を参照してください。

語彙を増やす最良の方法は辞書を引くことです。単語の発音→意味→用例の順に目を通し、派生語・類義語・対義語があればそれらを確認します。多義語については基本の意味から他の意味を導き出せるようにしてください。ある程度語彙力がついてきたら英英辞典も引くようにしましょう。

電子辞書や[英辞郎](#)のようなオンライン辞書もありますが、次に挙げる理由から紙の英和辞典を1冊持っておきましょう。

- 複数の単語を関連付けて覚えやすい
- 一覧性が高いのでポイントが分かりやすい
- 語義・語法の説明が詳細で簡潔に書かれている

英和辞典については[英語関係者のための情報](#)を参照してください。

語彙力の強化には以下のような方法も効果的です。

- 画像検索で単語とイメージを関連づける
- 身の回りの物やその状態を英語で言えるように習慣づける
- [P-Study System](#) などのアプリケーションを利用する

日本語と同じように英語にも話し言葉と書き言葉があります。話し言葉には基本的な単語しか出てきません。大学入試レベルまでの単語を知っていれば日常会話で困ることはほとんどありません。ただし会話では句動詞が多用されるので、単語だけではなく句動詞も覚えておく必要があります。和製英語も正しい英語で表現できるようにしておきましょう。日本語の表現は婉曲的で抽象的ですが、英語の表現は直接的で具体的です。例えば「すみません」や「よろしくお願いします」といった表現をそのまま英語に置き換えることはできません。場面に応じて適切な英語で表現する必要があります。会話表現については[マイスキ英語/オンライン英会話コラム](#)などが参考になります。

2.5 英語発音の注意点

英語では母音と子音の数が日本語よりも多く、その発音は日本語と比べて複雑ですが曖昧でもあります。例えば behind[bəháind] の発音はビハインドだったりバハインドだったりします。辞書によっても同じ単語の発音の表記がまちまちだったりします。しかし発音が少し間違ってもアクセントが正しければネイティブの人に通じることはよくあります。つまり英語では発音の正しさよりも音のリズムを重視しているということです。辞書に記載されている発音はあくまでも目安として捉えてください。

2.5.1 注意すべき母音

日本人が注意すべき英語の母音は æ と ɑ です。その他の母音は日本語の母音に近いので特に注意することはありません。

英語ではアクセントのある a の音はそのほとんどが æ と発音されます。またアクセントのある o の音はそのほとんどが ɑ と発音されます。どちらもアと発音して問題ありません。両者の違いは発音するときの口の形です。æ を発音するときには口角を少し上げる感じでアといいま

す。ɑ を発音するときはオの口の形でアと発音します。æ の音はアに近い音に聞こえたりエに近い音に聞こえたりします。例えば bank[bænk] の発音はバンクだったりベンクだったりします。ɑ の音はアに近い音に聞こえたりオに近い音に聞こえたりします。例えば stop[stáp] の発音はスタップだったりストップだったりします。英語の音は曖昧なのでこのような音の違いにあまり神経質にならないでください。

2.5.2 注意すべき子音

英語には日本語にはない子音がいくつかあります。

f と v は上の前歯を下唇に軽く乗せてその間から息を強く押し出します。θ と ð は舌尖に上の前歯を軽く乗せてその間から息を強く押し出します。英語の摩擦音は自分の耳に聞こえるくらいの鋭い音を少し長めに出すようにしてください。θ と ð には上の前歯の裏を舌尖で弾いて t や d に近い音を出す方法もあります。例えば Thank you[θ æŋkjú:] の発音はセンキューだったりテンキューだったりします。think[θ ínk] の発音もシンクだったりティンクだったりします。

r は舌を少し巻いて喉を鳴らしますが舌が上あごに触れないように注意してください。l は舌を上あごに押し当てて舌尖が上の前歯の裏に触れている状態で喉を鳴らします。r に母音が続くともった感じのラリルレロに聞こえます。l に母音が続くとはっきりした感じのラリルレロに聞こえます。両者は日本人には区別しにくい音ですがネイティブの人は明確に区別しています。l は母音の前では上記のような発音になります (これをライト L といいます) が子音の前や末尾では口に力を入れずにウと発音します (これをダーク L といいます)。

n はヌを母音をつけずに発音します。n の後ろに b もしくは p が続くと n は m に変化します。n の後ろに k もしくは g が続くと n は ŋ に変化します。ŋ はハミングするときのように鼻を鳴らします。ŋ は母音のない鼻濁音になることもあります。鼻濁音とはンガ・ンギ・ング・ンゲ・ンゴという鼻にかけるガ行の音です。

語尾の s は前の音が有声音 (喉を鳴らす音) ならズ [z] と発音し無声音 (喉を鳴らさない音) ならス [s] と発音しますが、語尾では有声音と無声音の区別が曖昧になるので実際の発音はどちらでも構いません。

2.5.3 注意すべき発音の単語

the[ðə] の th を発音するときは上の前歯の裏を舌尖で弾いて d に近い音を出します。従って the の音はダ [də](母音の前ではディ [di]) に近い音になります。

ths[θ s] は ts[ts] と同じ要領で発音します (ツに近い音になります)。ただし months[mán θ s] の ths の実際の発音はツ [ts] もしくはス [s] になります。従って months はマンツ [mánt्स]

もしくはマンス [máns] と発音します。

width[wíd θ] は d の音を破裂させずに発音するのでウィッツのような音になります。with[wíð] との発音の違いに注意してください。breadth\ [bréd θ \] / midst\ [mídst \] も同様にブレッツ・ミットゥのように発音します。

girl[gó:rl] はガロウもしくはグロウと発音します。curl[kó:rl] も同様にカロウもしくはクロウと発音します。missile[mísl] はミソウと発音します。tunnel[tʌnl] はタノウと発音します。mole[móul] はモウオウと発音します。語尾の l はオウと発音すればいいです。

u[jə] の前にある l は (子音の前にあるので) ダーク L になります。u[jə] はユと発音します。従って value[væljú:] / volume[vóljʊm] / evaluate[ivæljueít] / cellular[séljələr] / soluble[sáljəbl] はそれぞれバウユ・ボウユム・イバウユエイト・セウユラー・ソウユボウと発音します。

temperature[témp(ə)rət'fʊə] は 2 番目の母音を省略するのが一般的です。従って発音はテンパラチャではなくテンプラチャのようになります。

reference[réf(ə)rəns] / conference[kámf(ə)rəns] / comfortable[kámf(ə)təbl] も同様にレファレンス・カンファレンス・カンファタボウではなくレフレンス・カンフレンス・カンフタボウのように発音します。

アメリカ英語では m の後の p の発音は省略されることがあります。例えば attempt[ətémt] / empty[émti] / symptom[símptəm] / temptation[temptéifən] はアテンプト・シンプトム・テンプテイションではなくアテムト・エムティ・シムトム・テムテイションのように発音します。

アメリカ英語では n の前後の t を破裂させないことがあります。mountain[máuntn] はマウントゥンではなくマウツン、cotton[kátən] はカットウンではなくカツンのように発音します。internet[íntərnèt] はインタネットではなくインナネット、interview[íntərvjù:] はインタビューではなくインナビュのように発音します。

3 情報セキュリティ

3.1 はじめに

PC の安全性と機密性を高めるための方策を列挙します。

3.2 システムのセキュリティ

- ユーザアカウントにパスワードを設定する。
- 自動再生機能を無効にする。

- ソフトウェアをアップデートし最新の状態に保つ。
- 稼働中の PC から離れるときはシステムをロックする。
- 出所の不明なプログラムやスクリプトを実行しない。
- オープンソースのソフトウェアを選択する。
- ポータブル版のソフトウェアを選択する。
- [Process Explorer](#) でプロセスの状態を監視する。
- [CCleaner](#) で履歴やキャッシュを削除する。^{*1}

3.3 通信のセキュリティ

- PC とモデムの間にルータを介在させる。
- 無線 LAN の通信を暗号化する。
- 暗号化された通信プロトコルを使用する。
- メールの CC と BCC を正しく使い分ける。^{*2}
- メールやファイルを [GnuPG](#) で署名・暗号化する。^{*3}

3.4 メディアのセキュリティ

- ハードディスクのパーティションを [VeraCrypt](#) や [LUKS](#) で暗号化する。^{*4}
- リムーバブルメディアのデータを VeraCrypt のコンテナファイルに格納する。^{*5}
- メディアを廃棄するときは記録したデータを完全に消去し^{*6}メディアを物理的に破壊する。

^{*1} 詳細については [CCleaner](#) のダウンロードと使い方 - k 本的に無料ソフト・フリーソフトを参照してください。

^{*2} CC では送り先として指定したメールアドレスが相手に表示されますが BCC では表示されません。

^{*3} 秘密にする鍵と公開する鍵をペアで使用します。公開鍵で暗号化したデータは秘密鍵でしか復号できません。秘密鍵で署名したデータは公開鍵で検証できます。詳細については [GnuPG - ArchWiki](#) を参照してください。

^{*4} 詳細については [VeraCrypt](#) のダウンロードと使い方 - k 本的に無料ソフト・フリーソフトおよび [dm-crypt - ArchWiki](#) を参照してください。

^{*5} トラベラーズディスク (VeraCrypt のポータブル版) をコンテナファイルと同じ場所に作成しておくと便利です。

^{*6} ハードディスクのデータを削除するには Linux をライブ CD から起動してデータを上書きするコマンドを実行します。詳細については [ディスクの完全消去 - ArchWiki](#) を参照してください。

4 フォントとアイコン

4.1 はじめに

アプリケーションで利用できるフォントとアイコンを紹介します。

4.2 フォント

フォントには文字間隔が調整されるプロポーショナルフォントと文字幅が固定されている等幅フォントがあります。Windows には和文フォントとしてプロポーショナルフォントのMS Pゴシック・MS P明朝および等幅フォントのMS ゴシック・MS 明朝が用意されています。

Windows の ClearType という機能を有効にすればシステム全体のフォントを滑らかにすることができます。DirectWrite(Windows の文字描画機能) に対応したアプリケーションであればアンチエイリアス (もしくはフォントスムージング) という設定を有効にすることでも同様の効果を得られます。Windows には ClearType に最適化されたフォントとして和文フォントのメイリオ (Meiryo) および欧文フォントの Segoe が用意されています。

インターフェイスに適したフォントとしては Meiryo UI/Segoe UI/MS UI Gothic があります。文字幅が小さく文字間が狭いので限られた領域に多くの文字を表示することができます。ディスプレイでの表示を前提にデザインされているので視認性にも優れています。

ネットに公開されているフォントをダウンロード・インストールして使用することもできます。

プロポーショナルフォント	VL ゴシック / 源真ゴシック
等幅フォント	Ricty Diminished / Myrica

[Firefox](#) と [Thunderbird](#) でフォントを滑らかにする方法を紹介します。手順は次のとおり。

Firefox	アドレス <code>about:config</code> に移動すると設定エディタが起動するので検索ボックスに <code>gfx.font_rendering.clear_type_params.rendering_mode</code> と入力してその設定値を初期値の-1 から 5 に変更する
Thunderbird	メニューから「オプション」を選択して「一般」の最下部にある「設定エディター」をクリックすると設定エディタが起動するので検索ボックスに <code>gfx.font_rendering.clear_type_params.rendering_mode</code> と入力してその設定値を初期値の-1 から 5 に変更する

4.3 アイコン

いろいろな場面で利用できる便利なアイコンが [Tango Icon Gallery - Tango Desktop Project](#) で公開されています。



5 ファイルとアプリケーション

5.1 はじめに

よく使用するファイルの種類・ファイルフォーマットおよびそれらのファイルを扱うアプリケーションについて説明します。

5.2 テキストファイルの種類

プレーンテキスト (*.txt) テキストデータのみで構成された純粋なテキストファイルです。

設定ファイル (*.ini, *.xml) アプリケーションの設定が記述されたテキストファイルです。

HTML[HyperText Markup Language](*.html, *.htm) ウェブページの内容が記述されたテキストファイルです。

CSS[Cascading Style Sheets](*.css) ウェブページの体裁が記述されたテキストファイルです。

5.3 アーカイブファイルのフォーマット

Windows では主に ZIP というアーカイブフォーマットが使用されます。他のアーカイブフォーマットを扱うことはほとんどありません。

ZIP[ジップ](*.zip) Windows の標準的なアーカイブフォーマットです。

LHA[エルエイチエー](*.lzh) 国産のアーカイブフォーマットです。かつてはよく使用されていました。

CAB[キャブ](*.cab) Microsoft が開発したアーカイブフォーマットです。高い圧縮率を誇ります。

RAR[ラー](*.rar) ZIP よりも圧縮率の高いアーカイブフォーマットです。一定サイズに分割したアーカイブファイルを作成できます。

7z[セブンゼット](*.7z) ZIP よりも圧縮率の高いアーカイブフォーマットです。Unicode に対応しています。

gzip[ジージップ], bzip2[ビージップツー](*.gz, *.tar.gz, *.bz2, *.tar.bz2) Unix という OS の標準的なアーカイブフォーマットです。Windows で使用される ZIP とは関係ありません。

5.4 イメージファイルのフォーマット

イメージファイルは**ベクタイメージ** (ベクトルイメージ) と**ラスターイメージ** (ビットマップイメージ) に大別できます。

ベクタイメージには画像を拡大・縮小しても劣化しないという特長があります。

PostScript[ポストスクリプト](*.ps) Adobe が開発したページ記述言語 (プリンタに印刷内容を指示するためのプログラミング言語) です。

EPS[Encapsulated PostScript](*.eps) Adobe が上記の PostScript を元に開発したベクタ形式のイメージフォーマットです。

PDF[Portable Document Format](*.pdf) Adobe が上記の PostScript を元に開発したドキュメントフォーマットです。

SVG[Scalable Vector Graphics](*.svg) ウェブの標準化を目指す W3C が開発したベクタ形式のイメージフォーマットです。

WMF[Windows Metafile](*.wmf, *.wmz) Microsoft が開発したベクタ形式のイメージフォーマットです。

EMF[Enhanced Metafile](*.emf, *.emz) 上記の WMF を拡張したベクタ形式のイメージフォーマットです。

XPS[XML Paper Specification](*.xps) Microsoft が上記の PDF に対抗して開発したドキュメントフォーマットです。

OXPS[Open XML Paper Specification](*.oxps) 上記の XPS を国際規格として標準化したド

キュメントフォーマットです。

ラスタイメージはさらに無圧縮・可逆圧縮・非可逆圧縮の3つのフォーマットに分類できます。

5.4.1 無圧縮のイメージフォーマット

無圧縮のイメージファイルは他と比べて最も大きなファイルサイズになりますが元の画質は劣化しません。

BMP[ビットマップ、Microsoft Windows Bitmap Image](*.bmp) Windows で使用される無圧縮のイメージフォーマットです。

5.4.2 可逆圧縮のイメージフォーマット

可逆圧縮されたイメージファイルのファイルサイズは、無圧縮のイメージファイルより小さくなり非可逆圧縮のイメージファイルより大きくなります。元の画質は劣化しません。

PNG[ピン・ピング、Portable Network Graphics](*.png) あらゆる画像の保存に適した可逆圧縮のイメージフォーマットです。下記の GIF に代わるイメージフォーマットとして開発されました。GIF と比較して圧縮率が高くフルカラーを扱えるという特長があります。

GIF[ジフ、Graphics Interchange Format](*.gif) イラストの保存に適した可逆圧縮のイメージフォーマットです。元の画像が 256 色以上であれば減色されてしまい画質が落ちます。代替フォーマットの PNG が開発されましたが今でもよく使用されています。

5.4.3 非可逆圧縮のイメージフォーマット

非可逆圧縮されたイメージファイルのファイルサイズは他と比べて最も小さくなりますが元の画質は劣化します。

JPEG[ジェイペグ、Joint Photographic Experts Group](*.jpg, *.jpeg, *.jpe, *.jfif, *.jfi, *.jif) 写真の保存に適した非可逆圧縮のイメージフォーマットです。アプリケーションによっては保存する際に品質 (0~100) やファイルサイズを指定することができます。元画像より高い品質を指定すると元のファイルサイズより大きくなってしまうことがあるので、元画像が JPEG であれば品質の値は 80 辺りが適当です。JPEG による圧縮を繰り返すと画質は劣化していくので編集集中の画像は PNG で保存するようにしましょう。

5.5 オーディオファイルのフォーマット

オーディオファイルは無圧縮・可逆圧縮・非可逆圧縮の3つのフォーマットに分類できます。

5.5.1 無圧縮のオーディオフォーマット

無圧縮のオーディオファイルは他と比べて最も大きなファイルサイズになりますが元の音質は劣化しません。

WAV, WAVE[ウエーブ、RIFF waveform Audio Format](*.wav) Windows で使用される無圧縮のオーディオフォーマットです。オーディオ CD や外部入力から取り込んだ音声データはこのフォーマットで保存されます。音声データはオーディオ CD と同じ PCM 方式で記録されます。ファイルサイズは大きいですが (1 分あたりおよそ 10MB) 特別なデコード処理を必要としないため効果音のような短い音を保存するために使用されます。

5.5.2 可逆圧縮のオーディオフォーマット

可逆圧縮されたオーディオファイルのファイルサイズは、無圧縮のオーディオファイルより小さくなり非可逆圧縮のオーディオファイルより大きくなります。元の音質は劣化しません。フォーマットはいくつかありますがどれを選択しても大きな違いはありません。圧縮率・エンコード速度・再生負荷などが選択の基準になるでしょう。

FLAC[フラック、Free Lossless Audio Codec](*.flac, *.fla) 代表的な可逆圧縮のオーディオフォーマットです。圧縮率はそれほど高くありませんが、エンコードとデコードが速く多くの環境で利用できます。

5.5.3 非可逆圧縮のオーディオフォーマット

非可逆圧縮されたオーディオファイルのファイルサイズは他と比べて最も小さくなりますが元の音質は劣化します。非可逆圧縮を繰り返すと音質は劣化していきます。音質と圧縮率はトレードオフの関係にあり、それらはビットレートの大きさによって決まります。ビットレートが大きいほどファイルサイズは大きくなりますが音質は良くなります。ビットレートが小さいほどファイルサイズは小さくなりますが音質は悪くなります。

MP3[エムピースリー、MPEG-1 Audio Layer-3](*.mp3) 非可逆圧縮オーディオフォーマットのデファクトスタンダードです。ほとんどの環境で利用できます。複数のエンコーダが

存在しますが [LAME](#)(レイム) の音質が高く評価されています。

AAC[Advanced Audio Coding](*.aac) MP3 の後継として開発された非可逆圧縮オーディオフォーマットです。圧縮率が高く低ビットレートでも高音質を保つことができます。MP3 と同様に広く普及しています。

Vorbis[ヴォルビス](*.ogg) パテントフリーの非可逆圧縮オーディオフォーマットです。圧縮率が高く低ビットレートでも高音質を保つことができます。複数のエンコーダが存在しますが [aoTuV](#)(アオツブ) の音質が高く評価されています。Vorbis ではビットレートの代わりにクオリティレベルという値を使用します。ビットレートとの対応は次のとおり。^{*7}

クオリティレベル	-2	-1	0	1	2	3	4	5	6	7	8	9	10
ビットレート [kbps]	32	45	64	80	96	112	128	160	192	224	256	320	500

WMA[Windows Media Audio](*.wma) Microsoft が MP3 の代替として開発した非可逆圧縮オーディオフォーマットです。性能的には AAC/Vorbis とほとんど同じですがあまり使用されていません。

同一ビットレートで比較した場合、AAC/Vorbis/WMA は MP3 よりもが圧縮率が高く音質もいいとされています。ビットレートの設定については悩ましい問題ですが、MP3 なら 160kbps 以上、他のフォーマットなら 128kbps 以上で原音との区別が難しくなるようです。

5.6 ビデオファイルのフォーマット

ビデオファイルは無圧縮・可逆圧縮・非可逆圧縮の3つのフォーマットに分類できます。無圧縮のビデオファイルは巨大なファイルサイズになるので取り扱うことはありません。

5.6.1 可逆圧縮のビデオフォーマット

可逆圧縮されたビデオファイルのファイルサイズは、無圧縮のビデオファイルより小さくなり非可逆圧縮のビデオファイルより大きくなります。元の画質は劣化しません。フォーマットはいくつかありますがどれを選択しても大きな違いはありません。圧縮率・エンコード速度・再生負荷などが選択の基準になるでしょう。

HuffYUV[ハフワイユーバイ](*.avi) 代表的な可逆圧縮のビデオフォーマットです。

^{*7} [Hydrogenaudio](#) で行われたリスニングテストでは 128kbps に相当するクオリティレベルとして 4.25 という値が使用されました。エンコーダの実装上の問題を考慮して名目値よりも若干高めに設定されたようです。

5.6.2 非可逆圧縮のビデオフォーマット

非可逆圧縮されたビデオファイルのファイルサイズは他と比べて最も小さくなりますが元の画質は劣化します。非可逆圧縮を繰り返すと画質は劣化していきます。画質と圧縮率はトレードオフの関係にあり、それらはビットレートの大きさによって決まります。ビットレートが大きいほどファイルサイズは大きくなりますが画質は良くなります。ビットレートが小さいほどファイルサイズは小さくなりますが画質は悪くなります。

AV1[AOMedia Video 1](*.mkv, *.webm) 最新の非可逆圧縮ビデオフォーマットです。下記の H.265/VP9 を凌ぐ高い圧縮率を誇ります。今後普及することが期待されます。

H.264/MPEG-4 AVC(*.mp4) 現在最もよく使用されている非可逆圧縮ビデオフォーマットです。低ビットレートでも高画質を保つことができます。AAC との組み合わせが一般的です。

H.265/HEVC(*.mp4) H.264 の後継として開発された非可逆圧縮ビデオフォーマットです。H.264 と比較して画質・圧縮率が向上しています。AV1 が登場したので普及する見込みはありません。

VP9(*.webm) Google が開発した非可逆圧縮ビデオフォーマットです。画質・圧縮率は H.265 と同等です。AV1 が登場したので普及する見込みはありません。

DivX[ディビックス](*.avi, *.divx, *.div) DivX という会社が開発した非可逆圧縮ビデオフォーマットです。H.264 と比較して画質は劣りますが再生負荷は低いです。

Xvid[エクシビッド、エックスビッド](*.avi) DivX の商用化に反対して開発された非可逆圧縮ビデオフォーマットです。性能は DivX と同じです。

WMV[Windows Media Video](*.wmv) Microsoft が開発した非可逆圧縮ビデオフォーマットです。性能的には H.264 とほとんど同じですがあまり使用されていません。WMA との組み合わせが一般的です。

MPEG-1(*.dat, *.mpg, *.mpeg, *.m1v) ビデオ CD などで使用されている低画質の非可逆圧縮ビデオフォーマットです。

MPEG-2(*.mpg) DVD やデジタル放送に使用されている非可逆圧縮ビデオフォーマットです。H.264 と比較して画質は劣りますが広く使用されています。

MPEG-4(*.mp4, *.mpg) H.264 の前身に当たる非可逆圧縮ビデオフォーマットです。現在は H.264 が広く使用されているため見かけることは少ないです。

Motion JPEG[モーションジェイペグ](*.avi, *.mov) イメージフォーマットの JPEG を使用した非可逆圧縮ビデオフォーマットです。再生負荷は低いですが画質・圧縮率は良くありません。

DV(*.avi) VHS テープに使用されていた非可逆圧縮ビデオフォーマットです。今はほとんど

使用されていません。

H.264 ではビットレートではなく品質 (0~51) を指定するのが一般的であり、この値が小さいほど高画質ですがその分ファイルサイズは大きくなります。既定値は 23 であり 18 から 28 の間が推奨されています。画質にこだわるのであれば 20 辺りが適当です。

5.7 コンテナファイルのフォーマット

画像・音声・動画は**コンテナファイル**に格納されます。イメージファイルの画像およびオーディオファイルの音声は専用のコンテナファイルに格納されます。例えば MP3 のコンテナファイルには MP3 の音声しか格納することができません。ビデオファイルでは動画と音声が多重化されて動画用のコンテナファイルに格納されます。^{*8}格納できる動画のビデオフォーマットおよび音声のオーディオフォーマットはコンテナフォーマット毎に決められています。上記のビデオフォーマット (圧縮方法) と下記のコンテナフォーマット (動画形式) を混同しないように注意してください。

MP4(*.mp4, *.m4a, *.m4p, *.m4b, *.m4r, *.m4v) 現在最もよく使用されているコンテナフォーマットです。H.264 と AAC の組み合わせが一般的です。

MKV[Matroska](*.mkv, *.mka, *.mks, *.mk3d) 最も多機能なコンテナフォーマットです。あらゆるビデオフォーマット・オーディオフォーマットに対応しています。字幕を表示したり、字幕・音声を多重化したりすることもできます。

OGM[Ogg Media](*.ogg) MKV の元になったコンテナフォーマットです。見かけることはほとんどありません。

WebM[ウェブエム](*.webm) Google が開発したウェブ用のコンテナフォーマットです。

AVI[Audio Video Interleave](*.avi) Windows 標準のコンテナフォーマットです。歴史が古く低機能ですが汎用性が高いのでよく使用されます。

WMV[Windows Media Video](*.wmv, *.asf) WMV 専用のコンテナフォーマットです。ファイルの内部構造は下記の ASF と同じです。WMV と WMA の組み合わせが一般的です。

ASF[Advanced Systems Format](*.asf, *.asx, *.wmv, *.wvx, *.wma, *.wax) AVI の後継として開発されたコンテナフォーマットです。高機能ですがあまり使用されていません。

FLV[Flash Video](*.flv) Adobe Flash Player がサポートしているビデオフォーマットの動画を格納できるコンテナフォーマットです。^{*9}

^{*8} 動画のみであれば無音の動画が再生されます。音声のみであればオーディオファイルとして扱われます。

^{*9} かつては Flash Player でウェブの動画を再生していましたが現在は Flash から HTML5 に移行しています。

F4V(*.f4v, *.f4p, *.f4a, *.f4b) FLV を H.264/AAC に特化させたコンテナフォーマットです。

MOV(*.mov, *.qt) Apple の標準コンテナフォーマットです。

MPEG2-PS(*.mpg, *.mpeg, *.m2p, *.m2ps) DVD に使用されているコンテナフォーマットです。オーサリングする際に使用される VOB(*.vob) もこれと同じコンテナフォーマットです。

MPEG2-TS(*.ts, *.mts, *.m2t, *.m2ts) Blu-ray やデジタル放送で使用されているコンテナフォーマットです。

5.8 テキストファイルを扱うアプリケーション

テキストファイルは[テキストエディタ](#)を使用すれば表示・作成・編集することができます。

5.9 アーカイブファイルを扱うアプリケーション

アーカイブファイルは[アーカイバ](#)を使用すれば作成・展開・プレビューすることができます。

5.10 イメージファイルを扱うアプリケーション

イメージファイルは[イメージビューア](#)を使用すれば画像を表示したり他のフォーマットに変換したりすることができます。画像を作成・編集するには[イメージエディタ](#)を使用します。PDF ファイルは [PDF ビューア](#)を使用すれば閲覧することができます。イメージエディタを使用すれば PDF ファイルを画像として取り込み編集することができます。

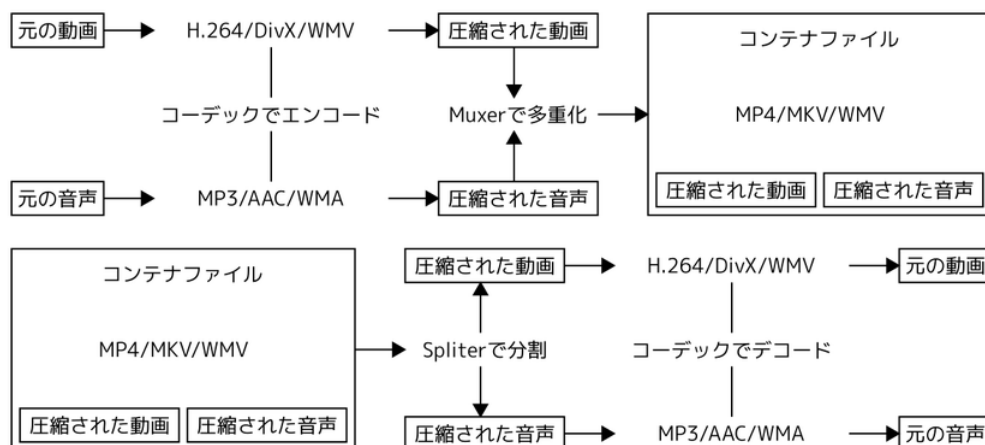
5.11 オーディオファイルを扱うアプリケーション

オーディオファイルは[オーディオプレーヤ](#)を使用すれば再生することができます。タグ (オーディオファイルに埋め込む曲情報) を編集したり他のフォーマットに変換したりするには [foobar2000](#) を使用します。foobar2000 があれば他のタグエディタやエンコーダは必要ありません。音声を編集するには[オーディオエディタ](#)を使用します。

従って今後 FLV が使用されることはありません。

5.12 ビデオファイルを扱うアプリケーション

ビデオファイルの再生にはビデオフォーマット・オーディオフォーマットに対応するコーデック (動画・音声を圧縮・復元するプログラム) が必要です。コーデックパックをインストールすればビデオプレーヤで問題なく再生することができます。コーデックを内蔵しているビデオプレーヤもあります。動画を編集したり他のフォーマットに変換したりするにはビデオエディタを使用します。ビデオファイルのエンコード・デコードのイメージは下のような感じです。



6 ワイルドカードと正規表現

6.1 はじめに

ワイルドカードおよび**正規表現**は文字列のパターンを指定する方法です。^{*10}パターンに文字列が一致することを「マッチする」といい、パターンを指定するための文字としてメタキャラクター (全て半角) が定義されています。

6.2 ワイルドカード

ワイルドカードに使用するメタキャラクターは "?" および "*" です。

"?" は任意の 1 文字を表します。

^{*10} ワイルドカードと正規表現を混同した記述が多く見受けられますが惑わされないよう注意してください。両者は似て非なるものです。

パターン "foo???" は "foo123", "fooabc" などの文字列にマッチします。

"*" は任意の文字列を表します。

パターン "foo*bar" は foo で始まり bar で終わる文字列にマッチします。

ワイルドカードはファイル名やファイルパスを指定するために使用されます。

6.3 正規表現

正規表現に使用するメタキャラクタは次のとおり。

`$ * + - . ? [\] ^ { | }`

これらのメタキャラクタはパターンの中で特別な意味を持ち他の文字とは区別されます。

メタキャラクタ以外の文字はその文字自身を表します。

パターン "foo" は文字列 "foo" にマッチします。

パターン "bar" は文字列 "bar" にマッチします。

メタキャラクタを普通の文字として扱いたいときは直前に "\" をつけます。これを「エスケープする」と言います。

パターン "foo\\bar" は文字列 "foo\bar" にマッチします。

パターン "foo*bar" は文字列 "foo*bar" にマッチします。

任意の文字を表すには "." と "[]" を使用します。

"." は改行を除く 1 文字を表します。

パターン "foo...bar" は "foofoobar", "foobarbar" などの文字列にマッチします。

"[]" 内の文字列はその文字列の各文字を表します。

パターン "[bar]" は "b", "a", "r" の各文字にマッチします。

"[]" 内に "\" を含める場合は "\\\" としてエスケープします。

パターン "[\\]" は文字 "\" にマッチします。

"[]" 内に "]" を含める場合は "\]" としてエスケープします。 "[" はそのまま問題ありません。

パターン "[\]" は文字 "]" にマッチします。

パターン "[[" は文字 "[" にマッチします。

"[]" 内の "-" は文字の範囲を指定するメタキャラクタになります。

パターン "[0-9]" は数字にマッチします。

パターン "[a-z]" は英小文字にマッチします。

パターン "[-bar]" は "-", "b", "a", "r" の各文字にマッチします。先頭の "-" はメタキャラクタとして扱われません。

パターン "[9-0]" は無効な表現です。文字コードが左から右へ昇順になるよう記述する必要があります。

"[]" 内の先頭の "^" は否定を意味するメタキャラクタになります。

パターン "[^bar]" は "b", "a", "r" 以外の文字にマッチします。

パターン "[^0-9]" は数字以外の文字にマッチします。

パターン "[bar^]" は "b", "a", "r", "^" の各文字にマッチします。先頭に位置しない "^" はメタキャラクタとして扱われません。

"[]" 内では "\", "]", "(先頭に位置しない)-", "(先頭の)^" がメタキャラクタとして扱われ、その他の文字はその文字自身を表します。

パターン "[.*]" は ".", "*" の各文字にマッチします。

パターンの連続を表すには "*", "+", "?", "{ }" を使用します。

"*" は直前のパターンの 0 回以上の連続を表します。

パターン "f*oo" は "oo", "foo", "ffoo", "ffffoo" などの文字列にマッチします。

パターン "ba*r" は "br", "bar", "baar", "baaaar" などの文字列にマッチします。

"+" は直前のパターンの 1 回以上の連続を表します。

パターン "f+oo" は "foo", "ffoo", "ffffoo" などの文字列にマッチします。

パターン "ba+r" は "bar", "baar", "baaaar" などの文字列にマッチします。

"?" は直前のパターンの 0 回以上 1 回以下の連続 (直前のパターンの有無) を表します。

パターン "f?oo" は "oo", "foo" の各文字列にマッチします。

パターン "ba?r" は "br", "bar" の各文字列にマッチします。

"{ }" は直前のパターンの指定回数の連続を表します。

{3}	3 回の連続
{3,}	3 回以上の連続
{3,4}	3 回以上 4 回以下の連続

パターン "f{3}oo" は文字列 "fffoo" にマッチします。

パターン "f{2,3}oo" は "ffoo", "fffoo" の各文字列にマッチします。

"*", "+", "?", "{ }" の直前にパターンが無い表現は無効です。

パターン "*foo", "+bar", "?foo" は全て無効な表現です。

メタキャラクターとして "*", "+" を含む表現は可能な限り長いパターンにマッチ (最長一致) します。可能な限り短いパターンにマッチ (最短一致) させるには "*", "+" の直後に "?" をつけます。

パターン ".*bar" は文字列 "foobarfoobar" に対して文字列 "foobarfoobar" にマッチします。

パターン ".*?bar" は文字列 "foobarfoobar" に対して文字列 "foobar" にマッチします。

位置を表すには "^" と "\$" を使用します。"^" は行頭を表します。"\$" は行末を表します。

パターン "^foo" は行頭にある文字列 "foo" にマッチします。

パターン "^.bar" は行頭から "bar" までの文字列にマッチします。

パターン "foo\$" は行末にある文字列 "foo" にマッチします。

パターン "^bar\$" は "bar" という文字列のみが存在する行にマッチします。

パターン "^^" は行頭にある文字 "^" にマッチします。行頭に位置しない "^" はメタキャラクターとして扱われません。

パターン "\$\$" は行末にある文字 "\$" にマッチします。行末に位置しない "\$" はメタキャラクターとして扱われません。

複数のパターンを指定するには "|" を使用します。

パターン `"foo|bar"` は `"foo"`, `"bar"` の各文字列にマッチします。

パターン `"foo|bar|foobar"` は `"foo"`, `"bar"`, `"foobar"` の各文字列にマッチします。

パターン `"f?oo|ba?r"` は `"oo"`, `"foo"`, `"br"`, `"bar"` の各文字列にマッチします。

メタキャラクターの適用範囲を指定するには `"()"` を使用します。

パターン `"(foo)?bar"` は `"bar"`, `"foobar"` の各文字列にマッチします。

パターン `"(foo)+bar"` は `"foobar"`, `"foofoobar"`, `"foofoofoobar"` などの文字列にマッチします。

パターン `"(foo|bar)foo"` は `"foofoo"`, `"barfoo"` の各文字列にマッチします。

`"\n"` は `n` 番目の `"()"` で囲まれたパターンを表します。

パターン `"(foo)(bar)\2\1"` は文字列 `"foobarbarfoo"` にマッチします。

慣用的な表現として以下のようなものがあります。

パターン `".*"` は任意の文字列を表します。(ワイルドカードの `"*"` と同じ意味になります)

パターン `"^$"` は空行を表します。

パターン `"^.*$"` は任意の 1 行を表します。

正規表現はテキストエディタなどで文字列の検索・置換に使用されます。

7 Windows のカスタマイズ

7.1 はじめに

Windows を使い始める前にやっておくべきことを紹介します。

7.2 パーティションの分割

Windows は C ドライブにインストールされます。従って Windows を再インストールすると C ドライブに作成したデータはすべて削除されてしまいます。このような場合に備えて C ドライブとは別のドライブにデータを保存するのが一般的です。

PC では物理的に 1 台しかないハードディスクを論理的に複数のハードディスクとして扱うことができます。具体的にはハードディスクを区画 (パーティション) に分割し、その区画にド

ライブを割り当てます。

Cドライブが占有しているハードディスクにDドライブを追加するには、Cドライブのパーティションを圧縮して空いたスペースに新しくパーティションを作成し、そのパーティションにDドライブを割り当てます。詳細については [Windows でパーティションを増やすには?](#)を参照してください。

7.3 パーティションの暗号化

Windows がインストールされている C ドライブ (が割り当てられているパーティション) をシステムドライブ (システムパーティション) といいます。情報漏洩を防ぐためにシステムドライブ以外のドライブのパーティションを [VeraCrypt](#) で暗号化します。暗号化したパーティションはパスワードを入力して任意のドライブにマウントします。詳細については [VeraCrypt のダウンロードと使い方 - k本的に無料ソフト・フリーソフト](#)を参照してください。

7.4 システムのカスタマイズ

7.4.1 視覚効果の無効化

1. スタートメニューから"コントロール パネル▷システムとセキュリティ▷システム▷システムの詳細設定"を選択します。
2. "システムのプロパティ"ダイアログの"パフォーマンス▷設定"をクリックします。
3. "視覚効果"タブから"パフォーマンスを優先する"をチェックして"OK"をクリックします。

7.4.2 自動再生の無効化

1. スタートメニューから"コントロール パネル▷ハードウェアとサウンド▷自動再生"を選択します。
2. "すべてのメディアとデバイスで自動再生を使う"のチェックを外して"保存"をクリックします。

7.4.3 常駐プログラムおよび自動起動の設定

1. スタートメニューの"プログラムとファイルの検索"に msconfig と入力して msconfig.exe を選択します。
2. "システム構成"ダイアログの以下を設定して"OK"をクリックします。

- "サービス"タブから Windows の稼動中に常駐させる必要のないプログラムのチェックを外します。
- "スタートアップ"タブから Windows 起動時に起動する必要のないプログラムのチェックを外します。

7.5 タスクバーのカスタマイズ

タスクバーは上に配置します。タスクバーの後ろにウィンドウが隠れてしまうのを防ぐには[助けてっば〜!](#)を使用します。

7.6 デスクトップのカスタマイズ

デスクトップにガジェットを置きたいなら [Rainmeter](#) を使用します。 [Rainmeter DeviantArt Gallery](#) などにスキンが公開されています。

7.7 キーバインドのカスタマイズ

[ChangeKey](#) でキーバインドを変更します。押しにくい位置にある使用頻度の高いキー (Ctrl キー・Tab キー・Esc キーなど) と押しやすい位置にある使用頻度の低いキー (CapsLock キー・Windows キー・アプリケーションキーなど) を入れ替えます。

7.8 アプリケーションのインストール

[窓の杜/k本的に無料ソフト・フリーソフト/FreeSoftNavi/フリーソフト 100](#)などを参考にし、自分に必要なアプリケーションをインストールします。どのアプリケーションをインストールしたらいいのか分からないのであればとりあえず[お気に入りのソフト](#)で紹介しているものを使ってみてください。

7.8.1 ポータブルアプリケーション

ポータブルアプリケーションとは USB メモリなどのメディアに入れて持ち運べる文字通りポータブルなアプリケーションです。ポータブルアプリケーションであればインストール作業が不要であり、インストールフォルダをコピーするだけで設定も含めて簡単にバックアップすることができます。利用可能であればポータブル版のアプリケーションを選択すべきです。インストーラでインストールするアプリケーションでもポータブル版としてインストールする

ことができるものもあります。主要なアプリケーションのポータブル版は [PortableApps.com](https://portableapps.com) から入手することができます。

7.9 ウェブブラウザのカスタマイズ

[Edge/Chrome/Firefox](#) には自由に機能を追加することができます。

[uBlock](#) を追加すればウェブページに表示される広告を一掃することができます。インストールしたら次の手順に従って設定します。

1. [豆腐フィルタ](#) (日本用の外部フィルタ) を追加します。
2. ツールバーの uBlock のアイコンをクリックして"ダッシュボード"のアイコンをクリックします。
3. 外部フィルターの"My フィルター"と"豆腐フィルタ"を有効にします。SNS のボタンが不要であれば"Fanboy's Social Blocking List"を有効にします。その他のフィルタは無効にします。
4. "適用"をクリックします。

ウェブページにブロックしたいものがあれば右クリックメニューから"要素をブロック"を選択します。自動的にフィルタが作成されて"My フィルター"に登録することができます。ブロックしたくないウェブサイトは"ホワイトリスト"に登録します。

この他にもたくさんの拡張機能があるので自分に合ったものを探してみてください。

	Edge/Chrome	Firefox
ポップアップをブロックする	Popup Blocker	Popup Blocker
ウェブページの背景を暗くする	Dark Reader	Dark Background and Light Text
ブックマークを新しいタブで開く	Neater Bookmarks	Open bookmarks in new tab
マウスのドラッグ操作を拡張する	QuickDrag for Chrome	QuickDrag WE
リンクの文字列をコピーする	Copy Link Text	Copy Link Text
複数ファイルを一括ダウンロードする	DownThemAll!	DownThemAll!
英語の意味を調べる	Google Dictionary	Mouse Dictionary
ウェブページの文章を読み上げる	Read Aloud	Read Aloud
YouTube を使いやすくする	Enhancer for YouTube	Enhancer for YouTube

Firefox のメモリの消費量を抑制するには[メモリ解放・最適化ツール - k 本的に無料ソフト・フリーソフト](#)で紹介されているようなアプリケーションを使用します。

8 かな入力のカスタマイズ

8.1 はじめに

かなの配列・入力方式の 1 つ[月配列](#)を紹介します。

8.2 月配列の特長

月配列には次のような特長があります。

- かな入力時にホームポジションからほとんど手を動かす必要がありません。
- かな入力時に小指をほとんど使用しません。
- シフト操作時にホームポジションから手を動かす必要がありません
- シフト操作時にシフトキーを押し続ける必要がありません。

8.3 月配列の種類

月配列にはいろいろなバージョンがあります。ここでは 2-263 版と Ux 版を紹介します。

8.3.1 2-263 版

[2-263 版](#)は月配列のデファクトスタンダードです。33 個のキーに 63 文字が割り当てられています。シフトキーにはホームポジションの中指のキーを使用します。シフト操作時はシフトキーを押してから次のキーを押します。

8.3.2 Ux 版

[Ux 版](#)は 2-263 版を改良して入力効率をさらに高めています。32 個のキーに 90 文字が割り当てられています。シフトキーにはホームポジションの中指と薬指のキーを使用します。シフト操作の方法は 2-263 版と同じです。

8.4 月配列の導入

月配列の導入方法については以下を参照してください。

- 2-263 版: 「月」 - 中指シフト新 JIS 配列
- Ux 版: [日本語入力用キーボード配列「月配列」\(Ux 版\) のページ](#)

9 DSP のカスタマイズ

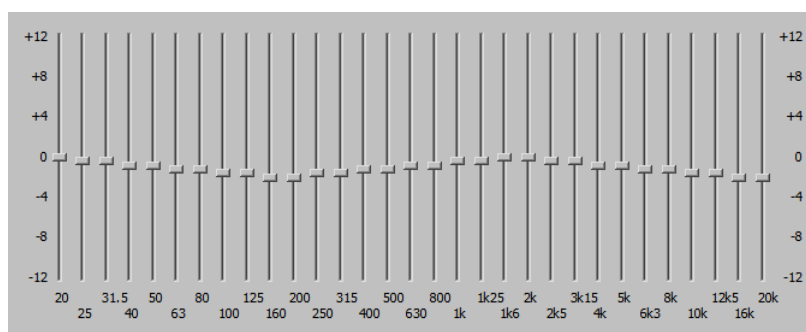
9.1 はじめに

オーディオプレーヤの DSP(再生音をフィルタリングする機能) について説明します。

9.2 DSP の機能

主な DSP としてイコライザ・コンプレッサ・リバーブがあります。

イコライザは音の高さに応じて音量を上げたり下げたりします。ここでは縦向きのスライドバーが横に並んだインターフェイスを有するグラフィックイコライザについて説明します。

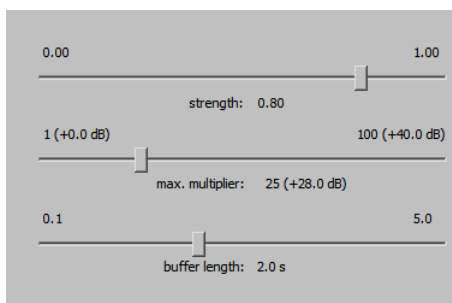


イコライザは低域・中域・高域 (インターフェースの左側・中央・右側) の 3つのパートに分けて調整します。各パートの音量を上げると音は次のように変化します。

- 低域の音量を適度に上げると丸く柔らかい音になるが、過度に上げるとこもった音になる
- 中域の音量を適度に上げると落ち着いた音になるが、過度に上げるとメリハリのない音になる
- 高域の音量を適度に上げると鋭く硬い音になるが、過度に上げると耳障りな音になる

9.2.1 コンプレッサ

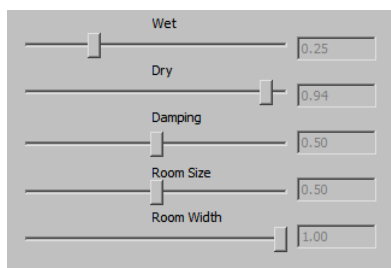
コンプレッサは小さすぎる音を大きくして大きすぎる音を小さくします。このような処理を「音圧を上げる」といいます。音圧を上げると迫力のある音になり、音量に大きな差のある曲が聴きやすくなります。しかし音圧を上げすぎると、音に歪みが生じたり曲が平板になったりします。



コンプレッサには音圧の強度や音量の増幅率を設定するパラメータがあります。曲全体の音量差を小さくするには音圧の強度を上げます。

9.2.2 リバーブ

リバーブは残響音を再現します。



リバーブには次のようなパラメータがあります。

Wet	Dry	Damping	Room Size	Room Width
残響の度合い	原音の度合い	反響しにくさ	空間の広さ	残響のステレオ感

9.3 DSP の使用例

9.3.1 XMPlay の DSP

[XMPlay](#) はイコライザ・コンプレッサ・リバーブの機能を標準装備しています。

DSP の設定方法は次のとおり。

1. XMPlay を起動します。
2. "Options and stuff"のボタンをクリックします。
3. 表示されたダイアログの左のツリーから"DSP"を選択します。
 - イコライザを有効にするには"Equalizer"をチェックします。右上のリストからプリセットを選択できます。
 - コンプレッサを有効にするには"Amplification"の"Auto-amp"から"dynamic"を選択します。"Reset on new track"のチェックははずしておきます。
 - リバーブを有効にするには"Reverb"をチェックします。time(残響時間)/level(残響レベル)/cutoff(反響しにくさ)を指定できます。
4. 閉じるボタンをクリックします。

9.3.2 foobar2000 の DSP

[foobar2000](#) にはイコライザ・コンプレッサ・リバーブ ([Graphic Equalizer](#)/[VLevel](#)/[Freeverb](#)) をプラグインで追加します。

DSP の設定方法は次のとおり。

1. foobar2000 を起動します。
2. メニューから"File▷Preference"を選択します。
3. 表示されたダイアログの左のツリーから"Components"を選択します。
4. "Install..."をクリックして上記のプラグインをインストールします。インストールが完了したら foobar2000 を再起動します。
5. 左のツリーから"Playback▷DSP Manager"を選択します。
6. 右の"Available DSPs"のリストから有効にしたい DSP の+をクリックすると"Active DSPs"のリストにプラグインが追加されます。ここでは VLevel, Graphic Equalizer, Freeverb DSP をこの順で有効にします。順番はドラッグで変更できます。× をクリックするとリストからプラグインが削除されます。
7. 左の"Active DSPs"のリストからそれぞれの DSP の"..."をクリックします。各 DSP

の設定ダイアログが表示されるので逐一設定します。

8. "OK"をクリックします。

10 ファイルのバックアップ

10.1 はじめに

ファイルをバックアップする方法を紹介します。

10.2 バックアップの準備

10.2.1 ハードディスク・SSD の構成

ここでは以下のようなハードディスク・SSD の構成を想定します。

内蔵のハードディスクもしくは SSD	C ドライブおよび D ドライブ
外付けのハードディスク	E ドライブ

C ドライブにインストールした Windows の再インストールに備えて C ドライブから D ドライブにファイルをバックアップします。自身が作成・ダウンロードしたファイルはすべて D ドライブに保存するようにするといいいでしょう。さらに内蔵ハードディスクの故障に備えて D ドライブのファイルを外付けハードディスクの E ドライブにバックアップします。内蔵ハードディスクに C ドライブしかない場合は新たに D ドライブを作成してください。^{*11}

10.2.2 保存先・バックアップ先のフォルダ

D ドライブには以下のようなフォルダを作成してファイルを分類整理します。プロファイルフォルダのフォルダには D ドライブのフォルダを割り当てておきます。やり方については[フォルダーを新しい場所にリダイレクトする](#)を参照してください。

AppData	プロファイルフォルダに保存された設定ファイル (*.ini, *.xml など)
---------	--

^{*11} PC では物理的に 1 台しかないハードディスクを論理的に複数のハードディスクとして扱うことができます。具体的にはハードディスクを区画 (パーティション) に分割し、その区画にドライブを割り当てます。C ドライブが占有しているハードディスクに D ドライブを追加するには、C ドライブのパーティションを圧縮して空いたスペースに新しくパーティションを作成し、そのパーティションに D ドライブを割り当てます。やり方については [Windows でパーティションを増やすには?](#)を参照してください。

Program Files	デフォルトのインストールフォルダに保存された設定ファイル ^{*12}
Applications	ポータブルアプリケーションそれぞれのインストールフォルダ
Applications\Common	複数のアプリケーションが共用するプログラム・ライブラリ (*.dll) ^{*13}
Downloads	Windows のデフォルトのダウンロード先
Shortcuts	アプリケーションのショートカット
Bookmarks	ウェブブラウザのブックマーク
Documents	ドキュメントファイル (*.odt, *.pdf, *.txt など)
Images	イメージファイル (*.bmp, *.jpg, *.png など)
Audios	オーディオファイル (*.aac, *.mp3, *.wma など)
Videos	ビデオファイル (*.mkv, *.mp4, *.wmv など)
Archives	アーカイブファイル (*.7zip, *.zip など)
RegFiles	レジストリからエクスポートしたデータ (*.reg)
ISOs	CD/DVD などのイメージファイル (*.iso)

10.3 バックアップの実行

2つの異なるドライブやフォルダの内容を同期させることをミラーリングといいます。[Backup](#) を使用すれば簡単にミラーリングすることができます。

D ドライブを E ドライブにミラーリングするには次のようにします。

1. Backup を起動します。
2. ウィンドウ右の"新規追加"をクリックします。
3. 表示されたダイアログの左のツリーから"名前"を選択し"パターン名"にバックアップパターンの名称を入力します。

^{*12} D\Program Files には C\Program Files のインストールフォルダから必要な設定ファイルのみをコピーします。インストーラによってアプリケーションをインストールする場合はデフォルトのインストール先である C\Program Files にインストールするようにしてください。他の場所にインストールすると問題を起すことがあります。

^{*13} このフォルダにパスを通してどのアプリケーションからも参照できるようにしておきます。パスの設定方法については [PATH を環境変数に設定する方法](#)を参照してください。

4. 左のツリーの"バックアップ元"を選択し"追加"をクリックして"ファイル名/フォルダ名"に D:\ を指定し"OK"をクリックします。
5. 左のツリーの"バックアップ先"を選択し"追加"をクリックして"フォルダ名"に E:\ を指定し"OK"をクリックします。
6. 左のツリーの"除外 1"を選択しバックアップ先にコピーしたくないファイルやフォルダがあれば指定します。
7. 左のツリーの"除外 2"を選択しバックアップ先にシステム属性・隠し属性のファイル・フォルダをコピーするかどうかを指定します。コピーするファイルの上限サイズも必要であれば指定します。
8. "OK"をクリックします。
9. 追加したバックアップパターンをリストから選択しウィンドウ下部の"バックアップ開始"をクリックします。

上記の手順と同様にして他のバックアップパターンも追加します。ウィンドウ下部の"全パターン開始"をクリックするとリストの順にすべてのバックアップパターンが実行されます。

11 レジストリのバックアップ

11.1 はじめに

レジストリをバックアップする方法を紹介します。

11.2 レジストリ全体のバックアップ

レジストリ全体をバックアップするには復元ポイントを作成します。復元ポイントの作成については[復元ポイントを作成する](#)を参照してください。復元ポイントの作成によってバックアップしたレジストリを復元するにはシステムの復元を実行します。システムの復元については[システムの復元](#)を参照してください。

11.3 レジストリキーのバックアップ

特定のレジストリキーをバックアップするにはレジストリエディタを使用します。詳細については[レジストリをバックアップする](#)を参照してください。

ここではバッチファイルを使用して一度に複数のレジストリキーをバックアップする方法を紹介します。テキストエディタで次のようなバッチファイル (拡張子が*.bat のテキストファイ

ル)を作成してください。

```
@echo off
```

:: レジストリのバックアップ

:: 最初にバックアップ先のフォルダを指定します。

:: 次のコマンドでバックアップ先のフォルダに移動します。

```
:: cd /d "バックアップ先のフォルダのパス"
```

```
cd /d "D:\RegFiles"
```

:: 以下でレジストリキーをバックアップします。

:: 次のコマンドでレジストリキーをエクスポートします。

```
:: reg export "エクスポートするキーのパス" "レジストリファイル (*.reg) の名前" /y
```

:: - キーのパスはレジストリエディタからコピーします。

:: - 複数のキーをバックアップする場合は複数行にわたって記述します。

```
reg export "HKEY_CURRENT_USER\Console" "Console.reg" /y
reg export "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\
Applets\Regedit" "Regedit.reg" /y
```

このバッチファイルを実行するとバックアップ先のフォルダにレジストリファイル (*.reg) が作成されます。このレジストリファイルを実行するとバックアップしたレジストリを復元することができます。

12 TeX と文書作成

12.1 はじめに

TeX(テフもしくはテック) は高度な組版作業を自動化します。TeX を利用すればワープロよりも効率的に文書を作成することができます。

12.2 T_EX の利点

ワープロと違い T_EX による文書作成ではフォントのサイズやスペースのとり方などを気にする必要がありません。文書の見た目についてはすべて T_EX が面倒を見てくれます。^{*14}精緻に組版されるので印刷の仕上がりはとてもきれいです。見出しや箇条書きの連番を割り振ってくれたり、目次や索引を生成してくれたりもします。書誌情報を管理する機能もあります。

12.3 T_EX の導入

T_EX を使用するために必要なものは以下の 3 つです。^{*15}

T _E X ディストリビューション	TeX Live
T _E X エディタ	LyX
PDF ビューア	Sumatra PDF

T_EX Live には組版に必要なプログラムがすべて含まれています。LyX は T_EX を使いやすくするためのエディタです。ワープロのようなインターフェイスで文書を作成・編集することができます。Sumatra PDF は組版処理された PDF 形式の文書を確認するために使用します。これらのアプリケーションのインストールや使い方の詳細については [TeX Wiki](#) を参照してください。

13 ローカルウィキ

13.1 はじめに

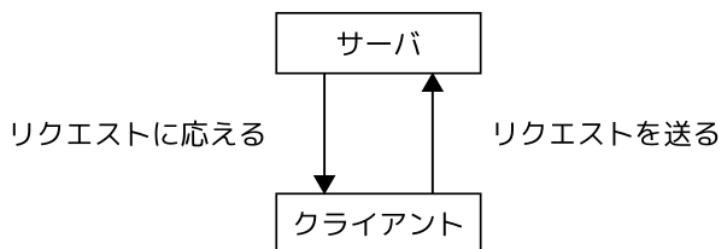
ウィキとはウェブブラウザからウェブサイトのページを作成・編集できる仕組みです。[HTML](#) よりもシンプルなマークアップ言語を使用して誰でも簡単にウェブページを作成することができます。

^{*14} ユーザが文書の見た目についてやるべきことは、あらかじめ用意された文書のクラスおよびスタイルを指定することだけです。文書の見た目を独自に調整することもできますが、その必要はほとんどありません。

^{*15} T_EX Live には [TeXworks](#) という T_EX エディタが含まれていますが T_EX のコマンドを直接記述する必要があるので初心者には不向きです。TeXworks は PDF ビューアとして使用することもできます。

13.2 サーバとクライアント

ウィキを使用する前にウェブの基本的な仕組みについて理解しておく必要があります。ウェブはサーバとクライアントのやりとりによって成り立っています。サーバはウェブなどのサービスを提供するプログラムです。クライアントはサーバにサービスを要求するウェブブラウザなどのプログラムです。クライアントはサーバにリクエストを送ります。サーバはクライアントからのリクエストを受け付けていて、リクエストに応じて保存されているウェブページなどのデータをクライアントに送り返します。このような仕組みによってウェブブラウザはウェブページを表示しています。従ってローカルウィキを利用するにはウェブブラウザからのアクセスに応えるためにローカルの環境でウェブサーバを稼働させておく必要があります。



13.3 Hugo

[Hugo](#) はウィキエンジンではなくサイトジェネレータですがローカルウィキのように利用することができます。Hugo の実行ファイル (hugo.exe) をインストールするだけですぐに使用することができます。下記の XAMPP よりも高速に動作します。ウェブサーバの機能を内蔵しているのでローカルな環境でもウェブサイトを表示することができます。マークアップ言語には [Markdown](#) を使用し、ショートコードという機能によってオリジナルのタグを作成することができます。使用方法については [まくまく Hugo/Go ノート](#) や [なかけんの Hugo ノート](#) を参照してください。

13.4 XAMPP/MediaWiki

[XAMPP](#) と [MediaWiki](#) を使用して Windows 環境にローカルウィキを構築する方法を紹介します。MediaWiki は [ウィキペディア](#) で使用されている高機能なウィキエンジンです。MediaWiki を使用するには以下のものがが必要です。

- ウェブサーバ (ブラウザからアクセスするため)
- データベースサーバ (コンテンツをデータベースで管理するため)
- PHP の実行環境 (PHP というプログラミング言語で書かれているため)

XAMPP は MediaWiki の使用に必要な上記のアプリケーションをまとめてインストールしてくれます。XAMPP をインストールすれば MediaWiki だけでなく他のウィキエンジンも使用することができます。自分に合ったウィキエンジンを探してみてください。

13.4.1 XAMPP のインストール

MediaWiki をインストールする前に MediaWiki に必要なアプリケーションを XAMPP でインストールします。XAMPP の公式サイト [Apache Friends](#) にはインストーラしかありませんがポータブル版も存在します。ここでは取り回しやすいポータブル版を選択します。[SourceForge](#) から最新の XAMPP(xampp-portable-win32-*.7z または xampp-portable-win32-*.zip) をダウンロードしてください。ダウンロードしたアーカイブを展開すると xampp というフォルダが生成されるのでこのフォルダをドライブの直下に置きます。他の場所に置くと動作しないので注意してください。^{*16}ここでは C ドライブに置くことにします。C:\xampp\xampp-control.exe を実行するとコントロールパネルが起動します。続けて Apache(ウェブサーバ) と MySQL(データベースサーバ) の"Start"をクリックします。Apache と MySQL が正常に起動したことを確認したら <http://localhost/> にアクセスします。"Welcome to XAMPP"というページが表示されたらインストール成功です。最後にコントロールパネルの"Config"をクリックして"Autostart of modules"の Apache と MySQL をチェックし"Save"をクリックします。次回からコントロールパネルを起動すると Apache と MySQL も同時に立ち上がります。C:\xampp\xampp-control.exe のショートカットをスタートアップフォルダに置けばすぐにウィキを使い始めることができます。

13.4.2 MediaWiki のインストール

XAMPP のインストールが終了したら MediaWiki をインストールします。インストールする前に MediaWiki で使用するデータベースを作成しておきます。Apache と MySQL を起動して [phpMyAdmin](#) にアクセスします。"データベース"をクリックしてデータベース名を入力し"作成"をクリックします。作成したデータベースの名前は MediaWiki のセットアップ作業で入力する必要があるので覚えておいてください。MediaWiki の公式サイト [MediaWiki](#) から MediaWiki をダウンロードします。ダウンロードしたアーカイブを展開すると mediawiki-*

^{*16} 他の場所に置いた場合は setup_xampp.bat を実行する必要があります。しかし不要な問題を避けるためドライブの直下に置くことを推奨します。

というフォルダが生成されます。このフォルダを mw にリネームして C:\xampp\htdocs に置きます。フォルダ名は任意ですがここでは mw とします。<http://localhost/mw/> にアクセスし指示に従って MediaWiki をセットアップします。セットアップが終了したらダウンロードした Localsetting.php を C:\xampp\htdocs\mw に置いて <http://localhost/mw/> にアクセスします。MediaWiki が正常に機能していることを確認してください。警告 (Warning) が表示される場合は C:\xampp\php\php.ini の

```
display_errors = On
```

を On から Off にします。

13.4.3 XAMPP のアップデート

XAMPP のアップデートは再インストールになります。新しい XAMPP をインストールする前に作成済みのデータベースをエクスポートします。Apache と MySQL を起動して [phpMyAdmin](#) にアクセスします。左のペインから自分が作成したデータベースをクリックします。"エクスポート"をクリックして"実行"をクリックします。"データベース名.sql"というファイルがダウンロードされます。複数のデータベースを作成した場合は同じ手順を繰り返します。エクスポートの作業が終了したら Apache と MySQL を停止して XAMPP を終了します。既存の C:\xampp をバックアップしてから新しい XAMPP をインストールします。インストールが終了したら C:\xampp\php\php.ini の以下の設定値を変更します。

```
max_execution_time = 300
memory_limit = 512M
post_max_size = 256M
upload_max_filesize = 256M
```

エクスポートしたデータベースをインポートするため Apache と MySQL を起動して [phpMyAdmin](#) にアクセスします。"データベース"をクリックしてエクスポートしたデータベースと同名のデータベースを作成します。データベース名を入力して"作成"をクリックします。続けて"インポート"をクリックします。"ファイル選択"をクリックしてエクスポートした同名のデータベースを選択し"実行"をクリックします。複数のデータベースをエクスポートした場合は同じ手順を繰り返します。最後にバックアップした xampp\htdocs\mw を C:\xampp\htdocs に戻します。

13.4.4 MediaWiki のアップデート

MediaWiki のアップデートは上書きインストールになります。C:\xampp\htdocs\mw のファイルを新しい MediaWiki のファイルで上書きしてください。続けて Apache と MySQL を起動しコマンドプロンプトで

```
C:\xampp\php\php.exe C:\xampp\htdocs\mw\maintenance\update.php
```

と入力します。上記のコマンドの処理が正常に終了したら <http://localhost/mw/> にアクセスします。MediaWiki が正常に機能していることを確認してください。

14 オーディオファイルの取り扱い

14.1 はじめに

[foobar2000](#) でオーディオファイルを取り扱う方法を紹介します。

14.2 オーディオファイルのノーマライズ

オーディオファイルの音量を均一化することをノーマライズといいます。ノーマライズするとファイルごとの音量の違いを意識する必要がなくなり、音量調節の煩わしさから解放されます。アルバムの曲をノーマライズする方法は次のとおり。

1. foobar2000 を起動します。
2. メニューから "File > Add files" または "Add folder" を選択しプレイリストにトラックを追加します。
3. 追加したトラックを選択して右クリックし "Replaygain > Scan Selection As Single Album" を選択します。
4. スキャンが終了したら "Update File Tags" をクリックします。この手順を省略したい場合はメニューから "File > Preference" を選択し表示されたダイアログの左のツリーから "Tools > ReplayGainScanner" を選択して "Quiet mode" をチェックし "OK" をクリックします。

上の処理でファイルの音声が増減されることはありません。ノーマライズの情報がタグとしてファイルに埋め込まれるだけです。

ノーマライズした音量で再生する方法はプレーヤごとに異なります。

ノーマライズした音量を [XMPlay](#) で有効にするには

1. XMPlay を起動します。
2. "Options and stuff"のボタンをクリックします。
3. 表示されたダイアログの左のツリーから"DSP"を選択します。
4. "Replaygain"から"Album"を選択し"pre-amp"を +8.0 程度にします。
5. 閉じるボタンをクリックします。

ノーマライズした音量を foobar2000 で有効にするには

1. foobar2000 を起動します。
2. メニューから"File▷Preference"を選択します。
3. 表示されたダイアログの左のツリーから"Tools▷Playback"を選択します。
4. "Replaygain"の"Source mode"を"album"、"Processing"を"apply gain"にします。
5. "Preamp"の"Without RG Info"の音量を-8.0dB 程度にします。
6. "OK"をクリックします。

ノーマライズすると音量が小さくなるため、ノーマライズされていないファイルは音量が相対的に大きくなります。両者の音量のバランスをとるためにプリアンプで音量を調整しています。

14.3 オーディオファイルのタギング

オーディオファイルには曲情報をタグとして埋め込む (タギングする) ことができます。

曲情報はデータベース (CDDb) から取得できます。取得方法は次のとおり。

1. foobar2000 を起動します。
2. 日本語の曲情報を取得するには日本語の CDDb サーバを登録します。この作業は次回以降は必要ありません。
 - (a) メニューから"File▷Preference"を選択します。
 - (b) 表示されたダイアログの左のツリーから"Tools▷Tagging▷freedb Tagger"を選択します。
 - (c) "Add"をクリックし表示されたダイアログに次のように入力して"OK"をクリックします。
 - Server: freedbtest.dyndns.org
 - Port: 80

- Address: /~cddb/cddb.cgi

(d) "OK"をクリックします。

3. メニューから"File▷Add files"または"Add folder"を選択しプレイリストにアルバムのトラックを追加します。
4. アルバムのトラックをトラック番号順にソートします。
5. アルバムのトラックを選択して右クリックし"Tagging▷Get Tags From freedb"を選択します。
6. 表示されたダイアログの曲情報を確認し"Update files"をクリックします。このとき取得した曲情報は自由に編集できます。曲情報が存在しなければその旨のメッセージが表示されます。

タグを編集するには次のようにします。

1. foobar2000 を起動します。
2. メニューから"File▷Add files または Add folder"を選択しプレイリストにトラックを追加します。
3. 追加したトラックを選択して右クリックし"Properties"を選択します。
4. 表示されたダイアログの"Value"の列をクリックしてタグを入力します。
5. "OK"をクリックします。

上の手順 3 で複数のトラックを選択すればそれらのタグをまとめて編集することができます。
例えばアルバムの曲をまとめてタギングするには次のようにします。

1. アルバムの曲を改行で区切ったテキストデータを作成しクリップボードにコピーします。
2. 上と同様にしてアルバムのトラックをプレイリストに追加します。
3. アルバムのトラックをトラック番号順にソートします。
4. アルバムのトラックを選択して右クリックし"Properties"を選択します。
5. 表示されたダイアログの"Track Title"をダブルクリックします。
6. 表示されたダイアログのリストアイテムをすべて選択し右クリックして"Paste"を選択します。
7. "OK"をクリックします。
8. "OK"をクリックします。

14.4 オーディオファイルのリネーム

タグを利用してファイルをリネームするには次のようにします。

1. foobar2000 を起動します。
2. メニューから "File▷Add files または Add folder" を選択しプレイリストにトラックを追加します。
3. 追加したトラックを選択して右クリックし "File Operations▷Rename to" を選択します。
4. 表示されたダイアログの "File Name Pattern" にリネーム後のファイル名を指定します。詳細については [TitleFormatting 解説 - foobar2000 Wiki](#) を参照してください。
5. "Preview" で処理内容を確認し "Run" をクリックします。

14.5 オーディオファイルのエンコード

オーディオファイルを圧縮 (エンコード) するには次のようにします。

14.5.1 エンコーダのインストール

エンコードに必要なソフト (エンコーダ) をダウンロード・インストールします。FLAC, MP3, Vorbis のエンコーダは [RareWares](#) から入手できます。以下のアーカイブをダウンロードしてください。^{*17}

- FLAC: Lossless▷FLAC v.1.3.3
- MP3: MP3▷LAME Bundles▷LAME 3.100
- Vorbis: Ogg Vorbis▷Oggenc▷Oggenc2.88

それぞれのアーカイブから flac.exe,lame.exe,oggenc2.exe を取り出し foobar2000 のインストールフォルダに置きます。

14.5.2 エンコーダの設定

エンコーダのパラメータを指定する場合はエンコードのプリセットを追加します。デフォルトのプリセットを使用するのであればこの作業は必要ありません。手順は次のとおり。

^{*17} エンコーダには 32bit 版と 64bit 版があります。自身の環境に合ったものを選択してください。

1. foobar2000 を起動します。
2. メニューから "File▷ Add files または Add folder" を選択しプレイリストにトラックを追加します。
3. トラックを選択して右クリックし "Convert" を選択します。
4. 表示されたダイアログの "Output format" をクリックします。
5. 表示されたダイアログの "Add New" をクリックします。
6. リストから "Custom" を選択し次のように設定して "OK" をクリックします。パラメータは必要に応じて変更してください。
 - FLAC
 - Encoder: flac.exe
 - Extension: flac
 - Parameters: -5 - -o %d
 - Format is: lossless
 - Highest BPS mode supported: 24
 - MP3
 - Encoder: lame.exe
 - Extension: mp3
 - Parameters: -b 160 - %d
 - Format is: lossy
 - Highest BPS mode supported: 24
 - Vorbis
 - Encoder: oggenc2.exe
 - Extension: ogg
 - Parameters: -q 4.25 - -o %d
 - Format is: lossy
 - Highest BPS mode supported: 32
7. 続けてプリセットを追加するなら手順 5 に戻ります。
8. "Cancel" をクリックします。

14.5.3 エンコードの実行

エンコードの手順は次のとおり。

1. foobar2000 を起動します。
2. メニューから "File ▷ Add files" または "Add folder" を選択しプレイリストにトラック

を追加します。

3. 追加したトラックを選択して右クリックし"Convert"を選択します。
4. 表示されたダイアログの"Output format"をクリックし出力ファイルのフォーマットを選択して"Back"をクリックします。
5. "Destination"をクリックし以下を指定して"Back"をクリックします。
 - "Output path"から出力ファイルの保存先を選択します。
 - Ask me later(後で尋ねる)
 - Source track folder(ソーストラックのフォルダ)
 - Specify folder(フォルダを指定する)
 - "If file already exists"から出力先に同名ファイルが存在するときの対処法を選択します。
 - Ask(尋ねる)
 - Skip(スキップする)
 - Overwrite(上書きする)
 - "Output style and file name formatting"からトラックをどのようにファイル出力するかを選択します。
 - Convert each track to an individual files(各トラックを個別のファイルに変換する)
 - Generate multi-track files(マルチトラックファイルを生成する)
 - Merge all tracks into one output files(すべてのトラックを1つの出力ファイルにマージする)
6. "Convert"をクリックします。

手順5の"Output style and file name formatting"について詳しく説明します。

通常は"Convert each track to an individual files"を選択し1トラックを1ファイルに保存します。"Name format"にはタグを利用したファイル名のフォーマットを指定できます。フォーマットの詳細については [TitleFormatting 解説 - foobar2000 Wiki](#) を参照してください。

"Generate multi-track files"は選択したトラックを結合して1つのファイルに保存し、トラックの頭出しに必要なキューシート (*.cue) というテキストファイルを出力します。"Name format & grouping pattern"にはファイル名のフォーマットおよびグルーピングのパターンを指定します。デフォルトの指定では出力ファイルがアルバムごとに分割されます。

"Merge all tracks into one output files"は選択したトラックを結合して1つのファイルに保存しますが、キューシートは出力されないのでトラックの頭出しはできません。

14.6 オーディオ CD のリッピング

オーディオ CD の曲を PC に取り込む (リッピングする) には次のようにします。

1. CD ドライブにリッピングしたい CD をセットして foobar2000 を起動します。
2. メニューから "File▷ Open audio CD" を選択します。
3. 表示されたダイアログのリストから使用する CD ドライブを選択します。
4. "Drive settings" をクリックし表示されたダイアログから CD ドライブを設定します。
同じ CD ドライブを使用する場合この作業は次回以降必要ありません。
(a) "auto" をクリックし "Read offset correction" の値を計測します。
(b) "OK" をクリックします。
5. "Rip" をクリックします。
6. CD 情報を取得したい場合は表示されたダイアログの "Information lookup" の "Source" から CDDDB (音楽 CD のデータベース) を選択し "Lookup" をクリックします。このとき取得した CD 情報は自由に編集できます。CD 情報が存在しなければその旨のメッセージが表示されます。日本語の CD 情報を取得したい場合は日本語の CDDDB を選択する必要があります。
7. "Proceed to the Converter Setup dialog" をクリックするとエンコードの設定ダイアログが表示されます。取得した CD 情報をタギングするために "Other" をクリックして "Transfer metadata (tags)" をチェックします。"ReplayGain-scan output files as albums" をチェックするとリッピング終了時にノーマライズされます。エンコードの設定が終了したら "Convert" をクリックします。"Rip now using one of the existing presets" をクリックすると下のリストから選択したプリセットを使用してリッピングされます。

リッピングが終了したら、リッピングした CD の曲がファイルとして出力先に取り込まれていることを確認してください。

15 Unix と Windows の違い

15.1 はじめに

パソコンを使用するには最初に OS というソフトをインストールする必要があります。市販のパソコンには Windows という OS が付属していますが Windows ではなく Unix 系の OS (FreeBSD や Linux) を使用することもできます。ここでは Unix を使用するにあたって

知っておきたい Windows との違いを紹介します。

15.2 ディストリビューション

様々なディストリビューション (配布形態) が存在しますが大きな違いはありません。多くのディストリビューションが無料で提供されています。

15.3 ユーザインターフェース

コマンドを直接入力するキーボード操作中心のインターフェース (CUI/CLI) です。

15.4 ユーザとグループ

root という名前のユーザがシステムの全権を掌握します。その他のユーザは一般ユーザとして扱われます。ユーザは複数のグループに所属できます。

15.5 ランレベルとデーモン

0～6 のランレベル (動作モード) を指定できます。ランレベルの一般的な定義は次のとおり。

0	停止	システムを停止もしくはシャットダウンする。
1	シングルユーザモード	root ユーザのみが作業できるシステムを起動する。ネットワークには接続せずデーモンも起動しない。
2～5	マルチユーザモード	デフォルトのシステムを起動する。
6	再起動	システムを再起動する。

ランレベル 1 が Windows のセーフモードに相当します。デーモンは Windows のサービスプログラムに相当します。

15.6 ブートローダ

ブートローダには主に GRUB (GRand Unified Bootloader) が使用されます。ブートローダはシステムを起動するプログラムです。Windows や他の OS とのマルチブートにも対応しています。

15.7 ファイルシステム

ファイルシステムには **UFS**(Unix File System) や **ext**(Extended File System) などが使用されます。Windows のファイルシステムのように断片化することはほとんどありません。**FAT** や **NTFS** などのファイルシステムにもアクセスすることができます。

15.7.1 ファイル

ファイル名の大文字と小文字は区別されます。ファイル名にスペースを入れることはできません。ファイル名の頭にピリオドをつけたファイル (通称ドットファイル) は隠しファイルとして扱われます。拡張子はユーザがファイルの種類を判別するために使用されますが、拡張子に基づいて動作するコマンドやアプリケーションもあります。Unix では拡張子のことをサフィクス (suffix) といいます。

15.7.2 ディレクトリ

ディレクトリは Windows のフォルダに相当します。

15.7.3 シンボリックリンク

シンボリックリンク (シmlink) は Windows のショートカットに相当します。

15.7.4 パーミッション

ファイルには読み出し/書き込み/実行の基本属性があります。実行属性が付与されていないプログラムやスクリプトは実行できません。ディレクトリもファイルと同じ属性を持ちますがその意味は次のようになります。

- ディレクトリの読み出し属性: ディレクトリのファイルリストを取得できる
- ディレクトリの書き込み属性: ディレクトリにファイルを作成できるまたはディレクトリのファイルを削除できる
- ディレクトリの実行属性: ディレクトリのファイルにアクセスできる

上記の基本属性に加えて SUID/SGID/スティッキーという特殊な属性があります。意味は次のとおり。

- 実行ファイルの SUID 属性: ファイルの所有者が実行したと見なされる

- 実行ファイルの SGID 属性: ファイルのグループに属するユーザが実行したと見なされる
- ディレクトリのスティッキー属性: ディレクトリのファイルを所有者以外のユーザが削除できなくなる

15.8 ディレクトリ構造

[FHS](#)(Filesystem Hierarchy Standard: ファイルシステム階層標準) で規定された[ルートディレクトリ](#)を頂点とするディレクトリ構造を持ちます。そのためシステムがインストールされるパーティションはルートパーティションと呼ばれます。Windows のようなドライブレターはなく他のドライブやパーティションは/dev に収められた[デバイスファイル](#)として参照されます。パスの区切り文字には\"\"(バックスラッシュ)ではなく\"/\"(スラッシュ)を使用します。詳細については [Man page of HIER](#) を参照してください。

15.9 ホームディレクトリ

/home に各ユーザのホームディレクトリが配置されます。ユーザを追加すると/etc/skel の内容が新しいユーザのホームディレクトリにコピーされます。コマンド入力時にはホームディレクトリのパスを~(チルダ) で置き換えることができます。

15.10 設定ファイル

システムやアプリケーションの設定はテキストファイルに保存されます。Windows の[レジストリ](#)のようなものは存在しません。設定を変更する場合はテキストエディタで設定ファイルを書き換えます。ユーザの設定はホームディレクトリにドットファイルとして保存されます。ファイル名のサフィクスとしてよく使用される rc は\"Run Command\"の略です。

15.11 アーカイブファイル

ファイルは [gzip](#)(* .gz) というプログラムで圧縮します。サイズの大きなファイルには gzip よりも圧縮率の高い [bzip2](#)(* .bz2) という圧縮プログラムを使用します。複数のファイルをアーカイブする場合は [tar](#)(* .tar) というプログラムで1つのファイルにまとめてから gzip や bzip2 で圧縮します。tar を使って圧縮されたファイル (* .tar.gz, * .tar.bz2) をターボール (tarball) ということがあります。紛らわしいですが Windows のアーカイブフォーマット (* .zip) とは関係ありません。

15.12 デバイスファイル

ハードディスクドライブや光学ドライブなどのデバイスはファイルとして扱われます。Windows のようにドライブレターで区別することはしません。デバイスファイルは `/dev` に収められています。デバイスにアクセスする場合はデバイスファイルを任意のディレクトリにマウントします。デバイスをディレクトリにマウントすると、マウントしたデバイスの内容をそのディレクトリから参照したり操作したりすることができます。`/etc/fstab` を書き換えて起動時に自動でマウントさせるようにすることもできます。

15.13 パッケージシステム

利用できるアプリケーションはパッケージという形でリポジトリというサーバに置かれています。パッケージの形式はディストリビューションによって異なります。パッケージはパッケージマネージャを使用してインストール・アップデート・アンインストールします。インストール済みのパッケージをまとめてアップデートすることもできます。

15.14 ウィンドウシステム

ウィンドウシステムには **X Window System** を採用しています。ディスプレイマネージャがログイン画面を表示してログインと同時に X が起動します。ログイン後はウィンドウマネージャがウィンドウを制御します。機能や見た目の異なるウィンドウマネージャを自由に選択することができます。

15.15 日本語入力システム

日本語入力は入力メソッドと変換エンジンによって実現されます。入力メソッドが入力を受け付け変換エンジンが漢字に変換します。

15.16 サウンドシステム

オーディオデバイスは **PulseAudio** や **ALSA**(Advanced Linux Sound Architecture) によって操作します。

15.17 印刷システム

プリンタは [CUPS](#)(Common Unix Printing System) によって操作します。

15.18 デスクトップ環境

Windows のようなデスクトップ環境も使用することができます。代表的なデスクトップ環境は [GNOME](#) と [KDE](#) です。より軽量のデスクトップ環境としては [Xfce](#) や [LXDE](#) などがあります。この他にも様々なデスクトップ環境が用意されています。

16 Unix の基本操作

16.1 はじめに

[MSYS2](#) を使用して Unix の基本操作を説明します。MSYS2 は Windows で動作するコンパクトな Unix 環境です。Unix については [Unix と Windows の違い](#) を参照してください。

16.2 MSYS2 のインストール

Windows のユーザ名に日本語が使用されている場合は MSYS2 のホームディレクトリのパスを環境変数 HOME に設定しておく必要があります。64bit 版では C:\msys64\home(32bit 版では C:\msys32\home) にホームディレクトリが作成されます。

```
HOME=C:\msys64\home\foo
```

[MSYS2 homepage](#) から MSYS2 のインストーラをダウンロードします。64bit 版 (msys2-x86_64-*.exe) と 32bit 版 (msys2-i686-*.exe) が置いてあるので自身の環境に合ったものを選択してください。

インストーラを実行しセットアップウィザードの指示に従ってインストールします。インストール先はデフォルトのまま C ドライブの直下にしておくのが無難です。パスにスペースなどが含まれていると問題が起きる可能性があります。

"Run MSYS2 now"をチェックしてセットアップウィザードを終了するとターミナル(コマンドの入出力を行うアプリケーション)が起動します。次のコマンドを入力して MSYS2 にインストールされているパッケージをアップデートします。


```
# pacman -Syu
```

以下のメッセージが表示されたらターミナルウィンドウの閉じるボタンをクリックします。

警告: terminate MSYS2 without returning to shell and check for updates again

警告: for example close your terminal window instead of calling exit

スタートメニューから MSYS2 を起動します。再度アップデートしてすべてのパッケージが最新であることを確認してください。

```
# pacman -Syu
```

16.3 ターミナル (mintty)

MSYS2 を起動すると root ユーザでログインした状態でターミナル [mintty](#) が起動します。MSYS2 ではすべての操作を root ユーザで行います。

16.3.1 コマンドラインの操作

ターミナルのコマンドラインは以下のようなキー操作で編集します。

Ctrl+j もしくは Ctrl+m もしくは Enter	入力したコマンドを実行
Ctrl+h もしくは BackSpace	カーソル位置の手前の文字を削除
Ctrl+d もしくは Del	カーソル位置の文字を削除
Ctrl+k	カーソル位置から行末の文字列を削除
Ctrl+u	カーソル位置から行頭の文字列を削除
Ctrl+f もしくは →	カーソルを右に移動
Ctrl+b もしくは ←	カーソルを左に移動
Ctrl+e	カーソルを行末に移動
Ctrl+a	カーソルを行頭に移動
Ctrl+i もしくは Tab	コマンド名・ファイル名・パス名を補完
Ctrl+n もしくは ↓	実行したコマンドの履歴を順方向に縦覧
Ctrl+p もしくは ↑	実行したコマンドの履歴を逆方向に縦覧

Ctrl+s	実行したコマンドの履歴を順方向に検索
Ctrl+r	実行したコマンドの履歴を逆方向に検索
Ctrl+g	コマンドの検索を終了
Alt+. もしくは Ctrl+[. もしくは Esc.	直前に実行したコマンドの最後の単語を挿入

ターミナルの画面出力を消去するには Ctrl+l を押します。

コマンド exit を入力するとログアウトしてターミナルが終了します。

```
# exit
```

16.3.2 ターミナルの設定

右クリックメニューから"Options"を選択するとターミナルの見た目や機能を細かく設定することができます。とりあえず表示されたダイアログの左のリストから"Window"を選択して"UI language"を"ja"にして"Save"をクリックします。これでインターフェースが日本語で表示されるようになります。

フォントと配色は大切なので最初に設定しておきましょう。設定ダイアログの左のリストから"テキスト"を選択するとフォントを指定することができます。見やすいフォントとしては [Ricty Diminished](#) や [Myrica](#) などがあります。

設定ダイアログの左のリストから"外観"を選択すると配色を指定することができます。配色のテーマ (*.minttyrc) は ~/.mintty/themes に置く と追加されます。 [mintty-color-schemes](#) のテーマを以下のようにして追加します。

```
# pacman -S git
# cd
# git clone https://github.com/oumu/mintty-color-schemes.git
# mkdir -pv .mintty/themes
# mv -v mintty-color-schemes/*.minttyrc .mintty/themes
# rm -rfv mintty-color-schemes
```

16.3.3 ロカールの設定

下のコマンドを入力すると現在のロカール (国・地域や言語の設定) が表示されます。

```
# echo $LANG
```

デフォルトのロカールは ja_JP.UTF-8 です。この意味は ja(言語は日本語)_JP(国は日本).UTF-8(使用する文字コードは UTF-8) です。ロカールの設定を変更する必要はありません。日本語入力についても設定することはありません。Windows と同じ方法で日本語を入力することができます。

16.3.4 マウント (fstab) の設定

起動時にデバイスやディレクトリをマウントするには下のように/etc/fstab を記述します。パスの区切りは"/"(スラッシュ) でも\"\"(バックスラッシュ) でも構いません。スペースは\040に置き換えます。/etc/fstab の設定を反映させるには MSYS2 を再起動する必要があります。

↓ /etc/fstab

```
C:\Users /users
C:\Program\040Files /pf
C:\Windows\System32 /sys32
```

16.3.5 man コマンドの日本語化

コマンドのヘルプを表示する man コマンドを日本語化するには次のようにします。

コマンド id を入力して自身のユーザ名と属しているグループ名を確認します。

```
# id
```

確認したユーザ名とグループ名を途中の質問で入力します。デフォルト (root) のままだとインストールに失敗します。

```
# cd
# wget https://linuxjm.osdn.jp/man-pages-ja-20191115.tar.gz
# tar xzvf man-pages-ja-20191115.tar.gz
# cd man-pages-ja-20191115
# make config
# make install
# mandb
# cd
# rm man-pages-ja-20191115.tar.gz
# rm -rfv man-pages-ja-20191115
```

16.4 パッケージマネージャ (pacman)

MSYS2 のパッケージマネージャは [pacman](#) です。詳細については [pacman - ArchWiki](#) を参照してください。

ヘルプを表示

```
# pacman -h
```

オペレーション-S のオプションを表示

```
# pacman -S -h
```

インストール済みのパッケージを表示

```
# pacman -Qqe
```

インストール済みのパッケージをアップデート

```
# pacman -Syu
```

パッケージグループのリストを表示

```
# pacman -Sg
```

パッケージグループ foo に含まれているパッケージを表示

```
# pacman -Sgg foo
```

名前に foo を含むパッケージを検索 (正規表現も使用可)

```
# pacman -Ss foo
```

```
# pacman -Sl | grep foo
```

パッケージ foo の詳細を確認

```
# pacman -Si foo
```

パッケージ foo をインストール

```
# pacman -S foo
```

パッケージ foo をアンインストール

```
# pacman -Rs foo
```

未使用のキャッシュを削除

```
# pacman -Sc
```

16.5 コマンドシェル (bash)

入力したコマンドはシェルによって実行されます。ここではデフォルトのシェル [bash](#) を使用します。詳細については [bash 入門](#)や [UNIX の部屋](#)を参照してください。

シェルの基本的なコマンドを以下に示します。

cat	ファイルの内容を表示、ファイルを連結
cd	カレントディレクトリを変更
chmod	ファイル・ディレクトリのパーミッション・属性を変更
chown	ファイルの所有者・グループを変更
cp	ファイル・ディレクトリをコピー
find	ファイル・ディレクトリを検索
grep	指定したパターンにマッチする行を検索
gzip, bzip2	ファイルを圧縮・展開
less	テキストファイルの内容を表示
ls	ディレクトリのファイルリストを表示
man	コマンドのマニュアルを表示
mkdir	新しいディレクトリを作成
mv	ファイル・ディレクトリを移動・リネーム
pwd	カレントディレクトリのパスを表示
rm	ファイル・ディレクトリを削除
tar	複数ファイルを単一ファイルにアーカイブ

実行中のコマンドを強制的に停止するには Ctrl+c を押します。

コマンドのオプションは-(ハイフン) で指定します。-(ハイフン) が単独で使用された場合は標準入出力として扱われます。スペース・タブ・改行はデリミタ (区切り文字) として機能します。

シェルには以下のメタキャラクタ (特殊文字) が存在しこれらの文字はシェルの中で特別な意味を持ちます。

" \$ & ' () * ; < > ? [\] ^ _ { | } ~

メタキャラクタを普通の文字として扱いたい場合は\ (バックスラッシュ) でエスケープします。シングルクォートで括れば文字列内のメタキャラクタをエスケープできます。

ワイルドカードを使用すればパターンにマッチする複数のファイルを一括で処理できます。詳細については[用語集: ファイルグロブ: UNIX/Linux の部屋](#)を参照してください。

パイプやリダイレクトを使用すればコマンドをつなげて実行したり入出力先を変更したりすることができます。詳細については[用語集: リダイレクト: UNIX/Linux の部屋](#)や[入力と出力 | UNIX & Linux コマンド・シェルスクリプト リファレンス](#)を参照してください。

リダイレクトについてかみ砕いて説明します。知っておくべきことは次のとおり。

- ディスクリプタは入出力先に割り当てられた番号
- 標準入力 (キーボード) のディスクリプタは 0
- 標準出力 (ディスプレイ) のディスクリプタは 1
- 標準エラー出力 (ディスプレイ) のディスクリプタは 2

ディスクリプタ n, m およびファイル foo に対して以下のように記述します。

n<foo	入力 n をファイル foo にセット (n=0 なら n を省略可)
n<&m	入力 n を入力 m にセット (n=0 なら n を省略可)
n>foo	出力 n をファイル foo にセット (n=1 なら n を省略可)
n>&m	出力 n を出力 m にセット (n=1 なら n を省略可)
&>foo もしくは >&foo	標準出力・標準エラー出力をまとめてファイル foo に出力

bash の設定は ~/.bashrc に記述します。変更した設定を有効にするには source(もしくは.(ドット)) コマンドを使用して設定ファイルを読み込む必要があります。

```
# source ~/.bashrc
# . ~/.bashrc
```

以下のようにコマンドのエイリアス (alias:別名) を定義することができます。

↓ ~/.bashrc

```
alias c='clear'
alias l='less -gmj10'
alias ls='ls -F --color=auto --group-directories-first'
alias la='ls -A'
alias ll='ls -Ahl --time-style=long-iso'
alias ..='cd ../'
alias ....='cd ../../'
alias cp='cp -iv'
alias mv='mv -iv'
alias rm='rm -iv'
alias mkdir='mkdir -v'
alias rmdir='rmdir -v'
alias chmod='chmod -v'
alias chown='chown -v'
alias vi='vim'
alias v='vim -R'
```

エイリアスを一時的に無効にしたい場合はコマンドの先頭に\`(バックスラッシュ)` をつけます。
例えばエイリアスとして設定した `ls` コマンドを本来の `ls` コマンドとして実行したいときは次のように入力します。

```
# \ls
```

以下のように関数を定義してコマンドを自作することもできます。

↓ ~/.bashrc

cd コマンドを実行すると ls コマンドを実行する

```
function cd {
    if builtin cd $1; then
        ls
    return 0
}
```

```

else
    echo "$1 に移動できません" 1>&2
    return 1
fi
}

```

一度にたくさんのコマンドを実行したい場合はシェルスクリプトを利用しましょう。詳細については[まぐまぐ Linux/Shell ノート](#)や[UNIX & Linux コマンド・シェルスクリプト リファレンス](#)を参照してください。

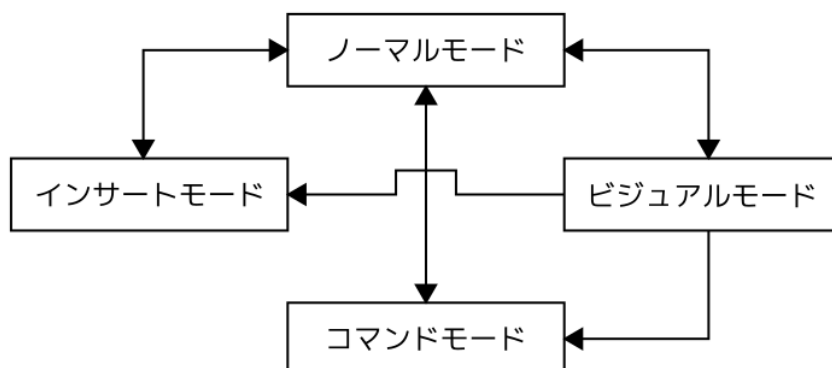
16.6 テキストエディタ (vim)

ここではテキストエディタとして [vim](#) を使用します。詳細については[まぐまぐ Vim ノート](#)や[名無しの vim 使い](#)を参照してください。

vim をインストールします。

```
# pacman -S vim
```

vim にはノーマル/インサート/ビジュアル/コマンドの4つのモードがあります。編集コマンドを受け付けるノーマルモードで起動し、ノーマルモードを軸に他の3つのモードに遷移します。文字を入力するときはインサートモードに移行します。範囲を選択するときはビジュアルモードに移行します。検索や置換などのコマンドを実行するときはコマンドモードに移行します。ノーマルモードもしくはビジュアルモードで:(コロン)を入力するとコマンドモードに移行し、下のコマンドラインから:に続けて任意のコマンドを入力することができます。ノーマルモードに戻るには Esc(Ctrl+[) もしくは Ctrl+c を入力します。



vim の基本的なコマンドを以下に示します。

ノーマルモード	
カーソルを下に移動	j もしくは Ctrl+n
カーソルを上に移動	k もしくは Ctrl+p
カーソルを右に移動	l もしくは Space
カーソルを左に移動	h もしくは BackSpace
カーソルを次の単語に移動	w もしくは W (記号を無視)
カーソルを次の単語の末尾に移動	e もしくは E (記号を無視)
カーソルを前の単語に移動	b もしくは B (記号を無視)
カーソルを前の単語の末尾に移動	ge もしくは gE (記号を無視)
カーソルを右方向・左方向の最初の j に移動	fj ↔ Fj
カーソルを右方向・左方向の最初の j の手前に移動	tj ↔ Tj
カーソルを行末に移動	\$
カーソルを行頭に移動	0 もしくは ^ (行頭の空白文字を無視)
カーソルを 128 行目に移動	128G
カーソルを最後の行に移動	G
カーソルを最初の行に移動	gg
画面を下にスクロール	Ctrl+f もしくは Ctrl+d (半画面)
画面を上スクロール	Ctrl+b もしくは Ctrl+u (半画面)
カーソル位置の文字を削除	x
カーソル位置の手前の文字を削除	X
カーソル位置の単語をカット	diw
カーソル位置の単語をコピー	yiw
カーソル位置から右側を削除	D
カーソル位置の行をカット	dd
カーソル位置の行をコピー	yy

カーソル位置 (行ならカーソル位置の下) にペースト	p
カーソル位置の手前 (行ならカーソル位置の上) にペースト	P
行を連結	gJ もしくは J (間にスペースを 1 文字挿入)
アンドゥ	u もしくは U (行全体)
リドゥ	Ctrl+r
直前の操作の繰り返し	.
カーソル位置の数字をインクリメント・デクリメント	Ctrl+a ↔ Ctrl+x
対応する括弧を検索	%
カーソル位置の文字を置換	r
カーソル位置の文字列を置換	R
ノーマルモード → インサートモード	
カーソル位置・カーソル位置の後ろから入力開始	i ↔ a
カーソル位置の行頭・行末から入力開始	I ↔ A
カーソル位置の下・上に挿入した空行から入力開始	o ↔ O
カーソル位置の文字・行を削除してから入力開始	s ↔ S
カーソル位置の単語を削除してから入力開始	ciw
カーソル位置の”“の中身を削除してから入力開始	ci”
カーソル位置から右側を削除して入力開始	C
ノーマルモード → ビジュアルモード	
カーソル位置から選択	v
カーソル位置から行選択	V
カーソル位置から矩形選択	Ctrl+v
直前の選択範囲を選択	gv
ビジュアルモード	
選択範囲をカット	d

選択範囲をコピー	y
選択範囲の大文字・小文字を反転	~
選択範囲を大文字・小文字に変換	U ↔ u
選択行をインデント・逆インデント	» ↔ «
ビジュアルモード → インサートモード	
選択範囲をカットして入力開始	c
インサートモード	
レジスタ”3”の内容をペースト	“3p
カーソル位置の上・下の行の文字をコピーして入力	Ctrl+y ↔ Ctrl+e
カーソル位置の行をインデント・逆インデント	Ctrl+t ↔ Ctrl+d
単語を順方向・逆方向に検索して補完	Ctrl+n ↔ Ctrl+p
コマンドモード	
変更を保存して終了	:x もしくは :wq もしくは ZZ
変更を保存せずに終了	:q! もしくは ZQ
レジスタの内容を表示	:di もしくは :reg
カーソル位置から文字列 foo を検索	/foo
カーソル位置の単語を検索	*
次の検索文字列・前の検索文字列に移動	n ↔ N
検索文字列のハイライトを解除	:noh
ファイル全体の文字列 foo を文字列 bar に置換	:%s/foo/bar/g
ファイル全体の文字列 foo を文字列 bar に確認しながら置換	:%s/foo/bar/gc
128 行から 256 行の文字列 foo を文字列 bar に置換	:128,256s/foo/bar/g
128 行から 256 行の文字列 foo を文字列 bar に確認しながら置換	:128,256s/foo/bar/gc
128 行から 256 行をアルファベットに順にソート	:128,256sort

ファイル foo をバッファに追加	:e foo
バッファのファイルをリスト	:ls
バッファのファイル foo に切り替え	:b foo
バッファのファイルをすべて終了	:qa
カーソル位置にファイル foo を挿入	:r foo

vim の設定は ~/.vimrc に記述します。以下の設定で使用するディレクトリ (~ / temp) はあらかじめ作成しておいてください。:so ~/.vimrc というコマンドを実行すると ~/.vimrc がリロードされます。so は source の略です。

↓ ~/.vimrc

```
set directory=$HOME/temp "スワップファイル (*.swp) を~/temp に保存
"set noswapfile "スワップファイルの機能を無効化
set backupdir=$HOME/temp "バックアップファイル (*) を~/temp に保存
"set nobackup "バックアップ機能を無効化
set undodir=$HOME/temp "アンドウの履歴を~/temp に保存
"set noundofile "アンドウの履歴を保存する機能を無効化
set number "行番号を表示
set laststatus=2 "ステータスラインを表示
set statusline=%F%m%h%w\ %<|\ENC:%{&fenc!=\'\'?&fenc:&enc}\|\FMT:%{&ff}\|\TYPE:%Y\|\
%=\|CODE:0x%02B\|\POS:%l,%02v\|
set whichwrap=b,s,<,>,[,] "矢印キーによる行移動を拡張
set background=dark "暗い背景色を指定
"set background=light "明るい背景色を指定
syntax on "シンタックスハイライトを有効化
colorscheme murphy "カラスキームを指定
set scrolloff=999 "カーソル位置の行を中央に表示
set expandtab "タブをタブ幅のスペースに置換
set autoindent "前の行のインデントを維持
set tabstop=2 "タブキーを押したときのタブ幅
set shiftwidth=0 "改行・タブコマンドを入力したときのタブ幅 (0 なら tabstop の値)
set shiftround "タブ幅の倍数になるように調整
set hlsearch "検索結果をハイライト
```

```
set incsearch "インクリメンタルサーチを有効化
set clipboard+=unnamed "OS のクリップボードと vim の無名レジスタを同期
set hidden "複数のバッファ（ファイル）を保持
```

"インサートモードで jj を入力するとノーマルモードへ移行

```
inoremap <silent> jj <ESC>
```

"インデント後に選択範囲を再選択

```
vnoremap < <gv
vnoremap > >gv
```

"全角スペースを表示

```
hi DoubleByteSpace term=underline ctermbg=blue guibg=darkgray
match DoubleByteSpace / /
```

"カーソル位置を復元

```
if has("autocmd")
au BufReadPost * if line("'\"") > 1 && line("'\"") <= line("$")
| exe "normal! g'\"" | endif
endif
```

vim のヘルプを日本語化するには次のようにします。

```
# pacman -S git
# cd
# git clone --depth 1 https://github.com/vim-jp/vimdoc-ja.git
# echo 'set helplang=ja,en' >>.vimrc
# echo 'set runtimepath+=$HOME/vimdoc-ja' >>.vimrc
# vim
```

vim が起動したら下のコマンドを実行します。

```
:helptags ~/vimdoc-ja/doc
```

ヘルプを表示するには:h と入力します。:q と入力すれば終了します。

テキストファイル~/foo を vim で開くには以下のように入力します。

```
# vim ~/foo
```

以下のように入力すると読み出しモードで開くことができます。

```
# vim -R ~/foo
```

```
# view ~/foo
```

1 から 10 の連番を入力するにはノーマルモードから `i` ▶ `0` ▶ `Enter` ▶ `9.` ▶ `ggVG` ▶ `g` ▶ `Ctrl+a` のようにします。

16.7 メールクライアント (mutt)

メールクライアントには [mutt](#) を使用することができます。詳細については [Mutt - ArchWiki](#) や [The Mutt E-Mail Client](#) を参照してください。

mutt をインストールします。

```
# pacman -S mutt
```

mutt の設定を `~/.muttrc` に記述します。

```
↓ ~/.muttrc
```

受信 (IMAP) の設定

```
set folder = imap[s]://imap.server.domain[:port]/[folder/] # IMAP サーバの  
アドレス
```

```
set imap_user = "USERNAME" # ユーザ名
```

```
set imap_pass = "PASSWORD" # パスワード
```

```
set spoolfile = +INBOX # 受信したメールを保存するメールボックス
```

```
mailboxes = +INBOX # 新着メールをチェックするメールボックス
```

```
set postponed = +Drafts # 下書きのメールを保存するメールボックス
```

```
set record = +Sent # 送信済みのメールを保存するメールボックス
```

```
set header_cache = ~/.cache/mutt # メッセージのヘッダを保存するディレクトリ
```

```
set message_cachedir = ~/.cache/mutt # メッセージを保存するディレクトリ
```

```
unset imap_passive # 新着メールをチェックする接続の許可
```

```
set imap_keepalive = 300 # 接続を維持するポーリングの間隔 (秒)
```

```
set mail_check = 120 # 新着メールをチェックする間隔 (秒)
```

送信の設定

```
set realname = "YOUR_REAL_NAME" # 名前
set from = "YOUR_E-MAIL_ADDRESS" # メールアドレス
set use_from = yes
set my_user="USERNAME" # ユーザ名
set my_pass="PASSWORD" # パスワード
set smtp_url=smtp[s]://$my_user:$my_pass@smtp.server.domain[:port] # SMTP
サーバのアドレス
# set ssl_force_tls = yes # SSL/TLS
# set ssl_starttls = yes # STARTTLS
set editor=vim # メッセージの編集に使用するエディタ（環境変数 EDITOR も使用可）
```

コマンド `mutt` を入力すると `mutt` が起動します。

```
# mutt
```

メールサーバに接続すると受信したメールがリストされます。選択して `Space` を押すとメッセージが表示されます。

メールボックスを変更するには `c` を押します。続けて `Tab` を押すとメールボックスがリストされます。選択して `Enter` を押すとメールボックスが変更されます。

メールを作成するには `m` を押します。宛先 (To) と件名 (Subject) を入力するとエディタが起動するのでメッセージを入力します。保存して `y` を押すとメールが送信されます。

16.8 ウェブブラウザ (w3m)

ウェブブラウザにはテキストブラウザの [w3m](#) を使用することができます。詳細については [w3m マニュアル](#) を参照してください。

`w3m` をインストールします。

```
# pacman -S w3m
```

URL を指定してコマンド `w3m` を入力すると `w3m` が起動しサイトが表示されます。

```
# w3m google.co.jp
```

基本的なキー操作は `less` や `vim` に似ています。マウス操作にも対応しています。

`H` を押すとヘルプが表示されます。設定を変更したいなら `o` を押します。`q` を押せば終了します。

キーバインドを変更したい場合は ~/.w3m/keymap を書き換えます。

次のような関数を定義すればコマンドからウェブを検索することができます。

↓ ~/.bashrc

ウェブを検索する

```
function sw {
    function display_message {
        echo 'DuckDuckGo:    sw "検索語句"' 1>&2
        echo '英辞郎:        sw e "検索語句"' 1>&2
        echo 'goo 国語辞書:   sw g "検索語句"' 1>&2
        echo 'e-Words:       sw ew "検索語句"' 1>&2
        echo 'Wikipedia:     sw w "検索語句"' 1>&2
    }
    if [ $# -eq 1 ]; then
        s='echo $1 | sed 's/ /+/g''
        w3m https://start.duckduckgo.com/?q=$s\&kl=jp-jp\&kp=-2\&kh=1\&k1=-1
        return 0
    elif [ $# -ne 2 ]; then
        display_message
        return 1
    fi
    case $1 in
        e )
            s='echo $2 | sed 's/ /+/g''
            w3m https://eow.alc.co.jp/search?q=$s
            ;;
        g )
            s='echo $2 | sed 's/ /+/g''
            w3m https://dictionary.goo.ne.jp/srch/jn/$s/m0u/
            ;;
        ew )
            s='echo $2 | sed 's/ /_/g''
            w3m http://e-words.jp/w/$s
            ;;
        w )
```



```

s='echo $2 | sed 's/ /_/g','
w3m https://ja.wikipedia.org/wiki/$s
;;
* )
    display_message
    return 1
;;
esac
return 0
}

```

16.9 ダウンロードマネージャ (curl, wget, aria2)

ダウンロードマネージャには [curl](#)/[wget](#)/[aria2](#) などを使用することができます。ここでは wget を紹介します。

wget をインストールします。

```
# pacman -S wget
```

ファイルをダウンロードするにはファイルの URL を指定してコマンド wget を入力します。

```
# wget http://www.example.com/path/to/file
```

ウェブサイトをダウンロードするには以下のオプションと共にウェブサイトの URL を指定してコマンドを入力します。毎回オプションを指定するのは面倒なのでエイリアスを設定しておくくと便利です。

```
# wget -m -p -E -k -np http://www.example.com/
```

16.10 FTP クライアント (lftp)

FTP クライアントには [lftp](#) を使用することができます。

lftp をインストールします。

```
# pacman -S lftp
```

コマンド `lftp` を入力すると `lftp` のシェルが起動して以下のようなプロンプトが表示されます。

```
# lftp
lftp :~>
```

FTP サーバのアカウントにログインするには続けて次のように入力します。USERNAME は自分のアカウントのユーザ名に置き換えてください。HOSTNAME は接続する FTP サーバのホスト名に置き換えてください。

```
lftp :~> open -u USERNAME HOSTNAME
```

続けてアカウントのパスワードを要求されます。正しいパスワードを入力するとログインに成功します。

FTP サーバにログインしたら `lftp` のコマンドを使用してサーバにファイルをアップロードしたりサーバからファイルをダウンロードしたりすることができます。`lftp` の基本的なコマンドには以下のようなものがあります。

<code>cd</code>	サーバのディレクトリを変更
<code>lcd</code>	クライアントのディレクトリを変更
<code>put</code>	クライアントのファイルをアップロード
<code>get</code>	サーバのファイルをダウンロード
<code>mput</code>	クライアントの複数ファイルをアップロード
<code>mget</code>	サーバの複数ファイルをダウンロード
<code>rm</code>	サーバのファイルを削除
<code>help</code>	コマンドリスト・コマンドのヘルプを表示

`lftp` を終了するにはコマンド `bye`/`exit`/`quit` などを入力します。

スクリプト (`lftp` で実行する一連のコマンドを記述したテキストファイル) を使用してクライアントからサーバにディレクトリの内容をミラーリングアップロードする方法を以下に紹介します。

あらかじめスクリプト `~/foo` を作成しておきます。

```
# vim ~/foo
```

↓ ~/foo

FTP サーバのアカウントにログインします。

ユーザ名 (USERNAME) ・パスワード (PASSWORD) ・ホスト名 (HOSTNAME) を自分のものに置き換えてください。

```
open -u USERNAME,PASSWORD HOSTNAME
```

クライアントからサーバにミラーリングアップロードします。

```
# mirror --reverse -R -n -e -p -v /local/path/to/dir /remote/path/to/dir
```

最初に以下のコマンドを実行して正しく処理されることを確認してください。

オプション"--dry-run"を外すと実際に処理されます。

```
mirror --reverse -R -n -e -p -v --dry-run /local/path/to/dir /remote/path/to/dir
```

ログアウトします。

```
close
```

スクリプト ~/foo を指定してコマンド lftp を入力します。

```
# lftp -v -f ~/foo
```

16.11 JavaScript シェル (V8)

JavaScript のプログラムを実行するには **V8** を使用します。

V8 をインストールします。

```
# pacman -S mingw-w64-i686-v8
```

```
# pacman -S mingw-w64-x86_64-v8
```

JavaScript のコードを記述したファイル ~/foo.js を作成します。

```
# vim ~/foo.js
```

↓ ~/foo.js

```
let a = 2;
```

```
let b = 3;
```

```
print( a + b ); // 5 (出力には print 関数を使用します)
```

~/foo.js を指定してコマンド d8 を入力します。

```
# d8 ~/foo.js
```

16.12 電卓 (bc)

電卓には [bc](#) を使用することができます。詳細については [bc コマンドの使い方: UNIX/Linux の部屋](#) を参照してください。

bc をインストールします。

```
# pacman -S bc
```

コマンド bc を入力すると bc が起動します。

```
# bc
```

```
bc 1.07.1
```

```
Copyright 1991-1994, 1997, 1998, 2000, 2004, 2006, 2008, 2012-2017
```

```
Free Software Foundation, Inc.
```

```
This is free software with ABSOLUTELY NO WARRANTY.
```

```
For details type 'warranty'.
```

続けて数式を入力すると計算結果が表示されます。終了するには quit と入力します。

```
2+3
```

```
5
```

```
quit
```

17 Linux のインストール

17.1 はじめに

[Arch Linux](#) を [VirtualBox](#) にインストールします。インストールするパッケージについては [ArchWiki](#) を参照してください。VirtualBox については [VirtualBox Mania](#) を参照してください。

17.2 BIOS/MBR と UEFI/GPT

Linux をインストールするにあたって BIOS/MBR と UEFI/GPT の違いを理解しておく必要があります。両者は機能的には同じもので UEFI/GPT は BIOS/MBR の後継として開発されました。

BIOS および UEFI には Windows などの OS とハードウェアの橋渡しをする役目があります。これらは PC にあらかじめ内蔵されたプログラムであり、電源投入時にハードウェアを初期化したり OS を起動したりします。

MBR および GPT はハードディスクなどのブロックデバイスの先頭にある小さな領域です。ここにはパーティション^{*18}の構成情報がパーティションテーブルとして格納されています。

OS をインストールすることができるパーティションをプライマリパーティションといいます。MBR では4つのプライマリパーティションしか作成できませんが^{*19}GPT では128のプライマリパーティションを作成することができます。

OS を起動するためのプログラムをブートローダといいます。ブートローダは電源投入時に BIOS もしくは UEFI によって実行されます。BIOS では MBR と特定のパーティションにブートローダが格納されています。UEFI では ESP という専用のパーティションにブートローダが格納されています。

BIOS

MBR	PP	PP	PP				EP
-----	----	----	----	--	--	--	----

UEFI

GPT	ESP	PP	PP	PP	PP	PP	PP
-----	-----	----	----	----	----	----	----

17.3 デバイスマッパーと LUKS/LVM

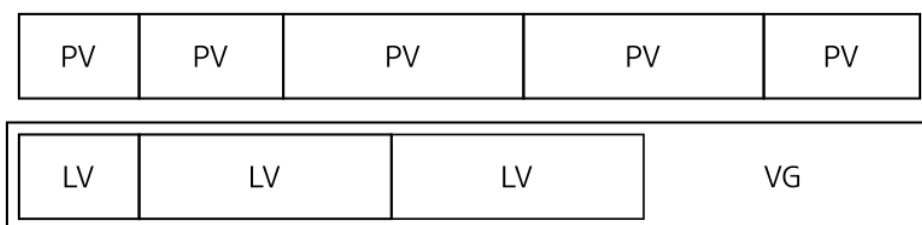
Linux にはデバイスマッパーと呼ばれるブロックデバイスの仮想レイヤーがあります。これは物理デバイスを抽象化して扱いやすくします。LUKS/LVM はこのデバイスマッパーを利用し

^{*18} ブロックデバイスは複数の区画に分割することができます。その区画をパーティションといいます。

^{*19} プライマリパーティションの1つを拡張パーティションに指定すればその中にいくつでもパーティションを作成することができます。ただし拡張パーティションには OS をインストールすることができません。

ています。LUKS はブロックデバイスを暗号化する仕組みです。LVM は複数の物理ブロックデバイスを仮想的な 1 つのブロックデバイスとして扱う仕組みです。

LVM には PV/VG/LV という 3 つの構成要素があります。PV は物理ブロックデバイスです。VG は PV で構成された仮想的な 1 つのブロックデバイスです。LV は VG に作成された仮想的なパーティションです。この仕組みによって物理ブロックデバイスの構成を気にすることなくパーティションを操作することができます。



17.4 Arch Linux のインストール

ここでは VirtualBox を使用して UEFI の仮想 PC に Arch Linux をインストールします。[Arch Linux のイメージファイル](#) (archlinux-*-x86_64.iso) をダウンロードしておいてください。

VirtualBox でインストール用に作成した仮想 PC を以下のように設定します。

1. VirtualBox マネージャーのリストから仮想 PC を選択して"設定"をクリックします。
2. 左のリストから"システム"を選択して"EFI を有効化"をチェックします。
3. 左のリストから"ストレージ"を選択して"ストレージデバイス"のリストから"空"と表示された CD のアイコンをクリックします。
4. 右上にある CD のアイコンをクリックしてメニューから"ディスクファイルを選択"を選択します。
5. 表示されたファイル選択ダイアログで Arch Linux のイメージファイルを選択して"開く"をクリックします。
6. 左のリストから"ネットワーク"を選択して"アダプター 1"の"ネットワークアダプターを有効化"がチェックされ"割り当て"が"NAT"になっていることを確認します。
7. "高度"をクリックして"ポートフォワーディング"をクリックします。
8. 一番右上のアイコンをクリックしてルールを追加し、ホストポートに 1024、ゲストポートに 22 を指定して"OK"をクリックします。

9. "OK"をクリックします。

インストール先のハードディスク (sda) のレイアウトを決めておきます。最初のパーティション (sda1) を ESP にします。次のパーティション (sda2) をルートパーティションにします。sda2 は [LUKS](#) で暗号化して、その上に [LVM](#) を構築します。

ESP	LVM: /dev/mapper/vg-root
/boot	LUKS: /dev/mapper/crypt-root
/dev/sda1	/dev/sda2

仮想 PC を起動してしばらくすると一時的な Unix 環境が構築され自動的に root ユーザにログインします。Arch Linux にはインストーラが用意されていません。シェルを使用してインストール作業を進めていきます。

キーボードレイアウトを日本語に設定します。

```
# loadkeys jp106
```

[ssh](#) で使用するためのパスワードを設定します。

```
# passwd
```

MSYS2 のターミナルで ssh を実行してホスト OS からゲスト OS に接続します。

```
$ ssh -p 1024 root@localhost
```

設定したパスワードを入力すれば接続できます。接続に失敗した場合は以下のようにして再度接続します。

```
$ ssh-keygen -R '[localhost]:1024'
```

以後は MSYS2 のターミナルで作業することができます。

UEFI モードで起動していることを確認します。

```
# ls /sys/firmware/efi/efivars
```

ネットワークインターフェイスが認識され有効であることを確認します。

```
# ip link
```

ここでは使用しませんが [Wi-Fi](#) に接続する方法を記載しておきます。ネットワークインターフェイスを wlan0 としています。最初にインターフェイス名とアクセスポイントとの接続状態を確認します。次にインターフェイスを初期化してアクセスポイント (SSID) と暗号化キー (PASSPHRASE) をインターフェイスに関連付けます。最後にインターフェイスに IP アドレスを割り当てます。

```
# iw dev
# iw dev wlan0 link
# ip link set wlan0 up
# wpa_supplicant -B -i wlan0 -c <(wpa_passphrase SSID PASSPHRASE)
# dhcpcd wlan0
```

インターネットに接続できていることを確認します。

```
# ping -c3 archlinux.jp
```

システムクロックを合わせて状態を確認します。

```
# timedatectl set-ntp true
# timedatectl status
```

ハードディスクが sda として認識されていることを確認します。

```
# lsblk
```

/dev/sda に [パーティション](#) を作成して正しくパーティショニングされていることを確認します。

```
# parted /dev/sda
(parted) mklabel gpt
(parted) mkpart ESP fat32 1MiB 551MiB
(parted) set 1 esp on
(parted) mkpart primary ext4 551MiB 100%
(parted) print
(parted) quit
```


sda2 を crypt-root というデバイス名で暗号化します。暗号化したデバイスやパーティションを使用するにはパスワードを入力して解錠する必要があります。解錠すると指定したデバイス名で/dev/mapper に追加され、暗号化したデバイスやパーティションを使用できるようになります。/dev/sda2 を解錠すると/dev/mapper/crypt-root として追加されます。

```
# cryptsetup luksFormat /dev/sda2
# cryptsetup luksDump /dev/sda2
# cryptsetup open /dev/sda2 crypt-root
```

crypt-root(暗号化した sda2) を PV(物理ブロックデバイス) として登録します。この PV を vg というグループ名の VG(仮想的なブロックデバイス) に追加します。そして vg 全体を使用して root というボリューム名の 1 つの LV(仮想的なパーティション) を作成します。作成した LV は"(VG 名)-(LV 名)"というデバイス名で/dev/mapper に追加されます。/dev/mapper/crypt-root に LVM が構築され、作成した LV が/dev/mapper/vg-root として使用できるようになります。このように LUKS の上に LVM を構築した場合、デバイスをまたいで LVM を使用することができなくなるので注意してください。

```
# pvcreate /dev/mapper/crypt-root
# vgcreate vg /dev/mapper/crypt-root
# lvcreate -l 100%FREE vg -n root
```

パーティションをフォーマットして/mnt にルートパーティションをマウントします。ESP は/mnt/boot にマウントします。マウントする順番を間違えないように注意してください。

```
# mkfs.vfat -F32 /dev/sda1
# mkfs.ext4 /dev/mapper/vg-root
# mount /dev/mapper/vg-root /mnt
# mkdir /mnt/boot
# mount /dev/sda1 /mnt/boot
```

パッケージは/etc/pacman.d/mirrorlist にリストされているミラーサーバからダウンロードされます。このリストに日本のサーバを追加しておきます。サーバはリストの上位にあるものが優先的に使用されます。

↓ /etc/pacman.d/mirrorlist

```
##
## Arch Linux repository mirrorlist
```

```
## Filtered by mirror score from mirror status page
```

```
## Generated on ****-**-**
```

```
##
```

```
Server = http://ftp.jaist.ac.jp/pub/Linux/ArchLinux/$repo/os/$arch
```

```
Server = http://ftp.tsukuba.wide.ad.jp/Linux/archlinux/$repo/os/$arch
```

ミラーサーバを変更したらパッケージのデータベースを更新します。このデータベースが最新の状態でなければパッケージのダウンロードに失敗するので注意してください。

```
# pacman -Syyu
```

システムの起動に必要な最小限のパッケージをインストールします。

```
# pacstrap /mnt linux base
```

[fstab](#) を作成して正しく書き出されていることを確認します。

```
# genfstab -U /mnt >>/mnt/etc/fstab
```

```
# cat /mnt/etc/fstab
```

インストールするシステムに [chroot](#) します。

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

タイムゾーンを設定します。

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

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

ロカールを設定します。`/etc/locale.gen` をエディタで開いて `en_US.UTF-8` および `ja_JP.UTF-8` のコメントを解除してください。

```
# pacman -S vim
```

```
# vim /etc/locale.gen
```

```
# locale-gen
```

```
# echo LANG=en_US.UTF-8 >/etc/locale.conf
```

コンソールのキーボードレイアウトおよび[フォント](#)を設定します。デフォルトのフォントは小

さいので大きめのフォントを指定しています。

```
# pacman -S terminus-font
# vim /etc/vconsole.conf
```

↓ /etc/vconsole.conf

```
KEYMAP=jp106
FONT=ter-v28b
```

ホスト名を追加します。以下の HOSTNAME は自分のホスト名に置き換えてください。

```
# echo HOSTNAME >/etc/hostname
# vim /etc/hosts
```

↓ /etc/hosts

```
# Static table lookup for hostnames.
# See hosts(5) for details.
127.0.0.1      localhost
::1           localhost
127.0.1.1     HOSTNAME.localdomain HOSTNAME
```

[DHCP](#) の設定を有効にします。

```
# pacman -S dhcpcd
# systemctl enable dhcpcd
```

[フック](#)を設定します。/etc/mkinitcpio.conf をエディタで開いて以下に示す箇所を書き換えます。項目の順番を間違えないように注意してください。

```
# vim /etc/mkinitcpio.conf
```

↓ /etc/mkinitcpio.conf

```
HOOKS=(... keyboard keymap block encrypt lvm2 ... filesystems ...)
```

LVM をインストールして初期 RAM ディスク ([initramfs](#)) を生成します。

```
# pacman -S lvm2
```

```
# mkinitcpio -p linux
```

root ユーザのパスワードを設定します。

```
# passwd
```

ブートマネージャ ([GRUB](#)) をインストールします。

```
# pacman -S grub efibootmgr
# grub-install --target=x86_64-efi --efi-directory=/boot --bootloader-id=grub
# grub-mkconfig -o /boot/grub/grub.cfg
# mkdir /boot/EFI/boot
# cp /boot/EFI/grub/grubx64.efi /boot/EFI/boot/bootx64.efi
```

/dev/sda2 の [UUID](#) を確認してから/etc/default/grub をエディタで開きます。起動時に暗号化した/dev/sda2 を解錠できるようにするためにカーネルのパラメータを設定します。デバイス名には UUID を指定してください。

```
# lsblk -f
# vim /etc/default/grub
```

↓ /etc/default/grub

```
GRUB_CMDLINE_LINUX="cryptdevice=UUID=81e69136-b1c6-46c7-8141-8e76554b2a2c
:crypt-root root=/dev/mapper/vg-root"
```

GRUB の設定を変更したら grub-mkconfig を実行して設定を更新する必要があります。忘れやすいので注意してください。

```
# grub-mkconfig -o /boot/grub/grub.cfg
```

chroot から抜けて/mnt にマウントしたパーティションをアンマウントします。最後に仮想 PC をシャットダウンします。

```
# exit
# umount -R /mnt
# shutdown -h now
```

仮想 PC から Arch Linux のインストールディスクを取り出します。

1. VirtualBox マネージャーのリストから仮想 PC を選択して"設定"をクリックします。
2. 左のリストから"ストレージ"を選択して右上にある CD のアイコンをクリックしメニューから"仮想ドライブからディスクを除去"を選択します。
3. "OK"をクリックします。

以上でインストール作業は終了です。Arch Linux が正常に起動することを確認してください。起動直後に LUKS で設定したパスワードを要求されます。パスワードが正しければルートパーティションが解錠されてシステムが起動します。

スワップファイルを作成します。スワップファイルはメモリが足りなくなったときに使用されます。メモリサイズに余裕があるのであれば必要ありません。再起動後に有効になっていることを確認します。

```
# fallocate -l 512M /swapfile
# chmod 600 /swapfile
# mkswap /swapfile
# swapon /swapfile
# echo /swapfile none swap defaults 0 0 >>/etc/fstab
# reboot
# free -h
```

VirtualBox の Guest Additions をインストールします。メニューの"デバイス"から"Guest Additions CD イメージの挿入"を選択してください。マウントする前に CD ドライブが sr0 として認識されていることを確認します。マウントしたら CD の内容を確認して VBoxLinuxAdditions.run というスクリプトを実行します。再起動後に有効になっていることを確認します。

```
# pacman -S linux-headers base-devel
# lsblk
# mkdir /mnt/cdrom
# mount /dev/sr0 /mnt/cdrom
# ls /mnt/cdrom
# sh /mnt/cdrom/VBoxLinuxAdditions.run
# reboot
# systemctl status vboxadd-service
```

一般ユーザを作成します。以下の USERNAME は自分のユーザ名に置き換えてください。

visudo で "%wheel ALL=(ALL) ALL" のコメントを解除してください。

```
# useradd -m -G wheel -s /bin/bash USERNAME
# passwd USERNAME
# EDITOR=vim visudo
```

root ユーザからログアウトして USERNAME でログインします。

```
# exit
```

pacman のカラー出力を有効にします。/etc/pacman.conf をエディタで開いて Color のコメントを解除してください。

```
$ sudo vim /etc/pacman.conf
```

使用されているグラフィックカードを確認した後、ビデオドライバおよび X をインストールします。次に X のキーボードレイアウトを日本語に設定します。最後に X が起動することを確認してください。

```
$ lspci | grep -e VGA -e 3D
$ sudo pacman -S xf86-video-vmware xorg-server xorg-apps xorg-xinit
  xorg-twm xorg-xclock xterm
$ localectl set-x11-keymap jp
$ startx
```

フォントをインストールします。ここでは Noto フォントをインストールしています。

```
$ sudo pacman -S noto-fonts noto-fonts-cjk noto-fonts-emoji noto-fonts-extra
```

フォントを設定します。/etc/fonts/local.conf に以下の設定を記述してフォントキャッシュを更新します。

```
$ sudo vim /etc/fonts/local.conf
```

```
↓ /etc/fonts/local.conf
```

```
<?xml version="1.0"?>
<!DOCTYPE fontconfig SYSTEM "fonts.dtd">
<fontconfig>
```

```

<alias>
  <family>serif</family>
  <prefer>
    <family>Noto Serif</family>
    <family>Noto Serif CJK JP</family>
  </prefer>
</alias>
<alias>
  <family>sans-serif</family>
  <prefer>
    <family>Noto Sans</family>
    <family>Noto Sans CJK JP</family>
  </prefer>
</alias>
<alias>
  <family>monospace</family>
  <prefer>
    <family>Noto Sans Mono</family>
    <family>Noto Sans Mono CJK JP</family>
  </prefer>
</alias>
</fontconfig>
$ fc-cache -fv

```

ロカールを日本語に設定します。

```
$ sudo vim /etc/locale.conf
```

```
↓ /etc/locale.conf
```

```
LANG=ja_JP.UTF-8
# LANG=en_US.UTF-8
```

ディスプレイマネージャ ([LightDM](#)) をインストールして有効にします。

```
$ sudo pacman -S lightdm lightdm-gtk-greeter
$ sudo systemctl enable lightdm
```

ネットワークマネージャ ([NetworkManager](#)) をインストールして有効にします。

```
$ sudo pacman -S networkmanager
$ sudo systemctl enable NetworkManager
$ sudo systemctl start NetworkManager
```

IME([fcitx5](#)) をインストールします。次に `~/.xprofile` をエディタで開いて以下の設定を記述します。最後に設定を開いて入力メソッドに `Mozc` が追加されていることを確認します。

```
$ sudo pacman -S fcitx5 fcitx5-mozc fcitx5-im fcitx5-configtool
$ vim ~/.xprofile
$ fcitx5-configtool
```

↓ `~/.xprofile`

```
export GTK_IM_MODULE=fcitx
export QT_IM_MODULE=fcitx
export XMODIFIERS=@im=fcitx
export DefaultIMModule=fcitx
fcitx-autostart
```

[AUR](#) のパッケージを扱えるようにするため AUR ヘルパー (`yay`) をインストールします。

```
$ sudo pacman -S git
$ git clone https://aur.archlinux.org/yay.git
$ cd yay
$ makepkg -si
$ cd
$ rm -rfv yay
```

デスクトップ環境 ([LXDE](#)) をインストールします。

```
$ sudo pacman -S lxde
```

再起動します。

```
$ sudo reboot
```


18 コーディングスタイル

18.1 はじめに

プログラミングにおいて読みやすく間違いに気づきやすいコードを記述するための工夫がコーディングスタイルです。以下に示すようなスタイルにこだわる必要はありませんが、一貫したスタイルで記述されていないコードにはやはり問題があります。ここでは参考として C++ のコーディングスタイルを紹介します。C++ のコーディングスタイルについては [C++ マニアック](#)や [wxWidgets のコーディングガイドライン](#)が参考になります。

18.2 コメント

他の人がコードを読む手助けになるようなコメントを記述するようにします。実行される文の処理をそのまま記述したようなコメントには意味がありません。

18.3 空白文字

18.3.1 空行

空行はコードを読みやすくするために使用します。コードブロックの前後に空行を挿入します。

18.3.2 スペース

スペースはコードを読みやすくするために使用します。括弧の前後や演算子の前後にはスペースを挿入します。

18.3.3 インデント

コードの階層構造を分かりやすくするために階層ごとにスペース 2 文字を行頭に挿入してインデントします。

18.4 識別子

変数や関数はコメントに頼らなくても役割や機能が分かるように命名します。プログラム全体で使用する定数は記号定数として定義します。そうすることによってコードが読みやすくなりコードのメンテナンスも容易になります。

識別子には意味のある英語を 3 単語以上使用し、それぞれの単語はできるだけ省略しないようにします。単語の区切り方には次のようなものがあります。

パスカルケース (PascalCase)	それぞれの単語の頭文字を大文字にする (先頭は大文字)
キャメルケース (camelCase)	それぞれの単語の頭文字を大文字にする (先頭は小文字)
スネークケース (snake_case)	それぞれの単語を"_"(アンダースコア)で区切る

変数名・関数名はパスカルケース (PascalCase) にします。定数名は大文字のスネークケース (SNAKE_CASE) にします。変数名には変数の型や種類を表すプレフィックスをつけます。ループのインデックスなど狭い範囲で使用する変数の名前はできるだけ短くして目立たないようにします。関数名は"述語 + 目的語"の形にして関数の機能が分かるようにします。戻り値のある関数の名前には戻り値の型を表すプレフィックスをつけます。

種類	プレフィックス	備考
クラス	C	Class
構造体	S	Structure
共用体	U	Union
列挙型	E	Enumerate
メンバ変数	m_	Member
グローバル変数	g_	Global
static	s_	
const	k	
[] (配列)	a	Array
& (参照)	r	Reference
* (ポインタ)	p	Pointer
*func() (関数ポインタ)	pf	Pointer of Function
テンプレート引数	t	Template

型	プレフィックス
---	---------

unsigned	u
short	s
long	l
bool	b
char	c
wchar_t	w
int	i
float	f
double	d
string	str
vector	vec
list	list
map	map
pair	pair
iterator	it

18.5 制御文

命令文が 1 行の場合でも "{ }" (ブレース) は省略しないようにします。命令文がない場合でも switch 文の default は省略しないようにします。

```

if ( ... ) {
    ...
} else {
    ...
}

if (false) {
} else if ( ... ) {
    ...
} else if ( ... ) {
    ...
} else {
    ...
}

if (false || ... || ... || ... ) {
    ...
}

if (true && ... && ... && ... ) {
    ...
}

```

```

switch ( ... ) {          for ( i = 0; i < 10; ++i ) {
break; case ... :          ...
    ...                    }
break; case ... :
    ...                    while ( ... ) {      do {
break; default:            ...                    ...
    ...                    }                    } while ( ... );
}

```

18.6 定義文

戻り値がなくても return 文は省略しないようにします。戻り値を"()" (括弧) でくくる必要はありません。

```
#include <string>
```

```
// 記号定数は名前空間の中に定義する
```

```

namespace symbol {
const int SIZE = 256;
} // namespace symbol

```

```

using namespace std;
using namespace symbol;

```

```

int i = 256;
int iYear, iMonth, iDay; // 相互に関連する変数はまとめて定義する
int aiMatrix[3][4] = { 11, 12, 13, 14, 21, 22, 23, 24, 31, 32, 33, 34, };

```

```

float fGetAverage(float f1, float f2) {
    return (f1 + f2) / 2.0;
}

```

```

class CDerived : public CBase {
public: // public メンバを先に記述する
    CDerived() : m_i(0) {

```

```

    ...
    return;
}
CDerived(const CDerived& r) { m_i = r.iGet(); return; }
~CDerived() {
    ...
    return;
}
int iGet() { return m_i; }
void Set(int i) { m_i = i; return; }

private:
    int m_i;
};

```

18.7 コードフォーマッタ (ClangFormat)

コードフォーマッタは設定に従ってプログラムのコードを整形してくれます。[Clang](#) というコンパイラに付属している ClangFormat は C 言語ライクなプログラミング言語 (C/C++/Objective-C/Objective-C++) に対応しているフォーマッタです。

Clang は MSYS2 でインストールすることができます。

```

# pacman -S mingw-w64-i686-clang
# pacman -S mingw-w64-x86_64-clang

```

整形したいソースファイルを指定してコマンド clang-format を入力します。

```

# clang-format -i -style=google foo.c

```

オプション-style でスタイルを指定します。オプション-i でファイルを上書きします。

以下のようなスクリプトを実行すればカレントディレクトリのソースファイルをまとめてフォーマットすることができます。

```

#!/bin/bash

for infile in *.{cpp,h}; do

```

```
echo clang-format -i -style=google $infile
clang-format -i -style=google $infile
done

exit 0
```

19 Windows プログラミング

19.1 はじめに

MSYS2 という開発環境を使用して Windows アプリケーションを作成する方法を紹介します。MSYS2 については [Unix の基本操作](#) を参照してください。

19.2 プログラミングの予備知識

以下に掲載するコードは [C 言語](#) および [C++](#) というプログラミング言語で記述されています。C 言語で記述されたコードの中では Windows の機能にアクセスするために [WindowsAPI](#) を使用しています。^{*20} WindowsAPI を使用するためのツールが WindowsSDK です。C 言語と WindowsSDK の組み合わせを「C と SDK」といったりします。

C++ は C 言語を拡張してオブジェクト指向でプログラミングできるようにした言語です。C++ は C 言語との互換性を持たせて拡張したので他の純粋なオブジェクト指向の言語と比べて仕様が複雑になっていますが C 言語にはない便利な機能がたくさん追加されています。詳細については [ゼロから学ぶ C++/Programming Place Plus/C++ マニアック](#)などを参照してください。

アプリケーションを作成するにはプログラムのソースコードを編集するためのテキストエディタをはじめ、ソースファイルからオブジェクトファイルを生成するためのコンパイラやオブジェクトファイルとライブラリをリンクするためのリンカなどが必要です。アプリケーションを作成するためのこれらのプログラムを総称して [ツールチェーン](#)といいます。

GUI のアプリケーションを作成するには [GUI ツールキット](#)と [GUI ビルダ](#)を使用します。WindowsAPI は Windows に内蔵されている低レベルなツールキットです。[WTL](#)や [wxWidgets](#) などの高レベルなツールキットを使用することもできます。詳細については [WisdomSoft](#)や [viva Cocoa](#)を参照してください。

^{*20} 独自の変数型や構造体がたくさん出てきますが難しく考える必要はありません。

19.2.1 ポインタ

C 言語のポインタについてかみ砕いて説明します。変数のデータはコンピュータのメモリのどこかに置かれています。その置かれている場所をアドレスといいます。アドレスは変数の住所のようなものだと考えてください。変数は変数名だけでなくアドレスによっても表現することができます。ポインタは変数のアドレスを格納するための変数です。ポインタと聞くと何か特別なもののように感じますがただの変数に過ぎません。ポインタ変数もしくはアドレス型の変数といったほうが分かりやすいかもしれません。

ポインタの値すなわち変数のアドレスを変更するためにポインタのポインタが使用されることがあります。ポインタのポインタと聞くと頭が混乱するかもしれませんが難しいことはありません。以下のコードを見てよく考えてください。

```
#include <stdio.h>

int main(void) {
    char *ptr;          // ポインタ
    char **ptrptr;      // ポインタのポインタ
    char str1[256] = "foo";
    char str2[256] = "bar";
    ptr = str1;          // ptr に str1 のアドレスを代入
    printf("%s\n", ptr); // foo と表示
    ptrptr = &ptr;       // ptr のアドレスを ptrptr に代入
    *ptrptr = str2;      // ptrptr を参照して ptr に str2 のアドレスを代入
    printf("%s\n", ptr); // bar と表示
    return 0;
}
```

ポインタに `const` を適用する場合、ポインタの示すアドレスを定数にするのかそれともそのアドレスに置かれているデータを定数にするのかという問題が生じます。これは次のように書き分けます。

```
int* const p; // アドレスが定数
const int* p; // データが定数
int const* p; // データが定数
const int* const p; // アドレス・データともに定数
```

```
int const* const p; // アドレス・データともに定数
```

19.2.2 クラスおよびオブジェクト

C++ のクラスについて簡単に説明します。C 言語の構造体はそのメンバとして型の異なる複数の変数を持つことができます。クラスはメンバとして関数を持てるように構造体を拡張したものです。クラスのメンバ変数をプロパティ、クラスのメンバ関数をメソッド、クラス型の変数をオブジェクト、クラスを変数として実体化することをインスタンス化といいます。

オブジェクトはデータに機能を持たせたものと捉えることができます。オブジェクトのプロパティはメソッドを介して参照・操作します。プロパティを直接参照したり操作したりすることは一般的ではありません。このようにインターフェイスのみを公開して実装を非公開にすることをカプセル化といいます。カプセル化することによってプログラムの保守性・拡張性・移植性を高めることができます。

C++ のクラスのメンバには `private/public/protected` という 3 つの属性があります。private メンバは自クラスからしか参照することができず外部からは参照することができません。public メンバは自クラスだけでなく外部からも参照することができます。protected メンバは private メンバと同様に外部からは参照することができませんが自クラスだけでなく派生クラスからも参照することができます。派生クラス (サブクラス) とは文字通りあるクラスから派生したクラスです。派生元のクラスを基底クラス (ベースクラス) といい派生クラスは基底クラスのメンバを継承します。メソッドに `virtual` というキーワードをつけるとそのメソッドを派生クラスで再定義 (オーバーライド) することができます。また C++ では引数の型や数の異なる同名の関数を定義 (オーバーロード) することができます。オーバーライド (メソッドの再定義) とオーバーロード (関数の多重定義) を混同しないように注意してください。

C 言語は文字列を扱うのが苦手ですが C++ には `string` というクラスが用意されているので文字列を簡単に処理することができます。データ構造やアルゴリズムを扱うクラスライブラリとして [STL](#) も用意されています。

19.3 アプリケーションの開発環境

実際にアプリケーションを作成する前にアプリケーションの開発環境を構築する必要があります。ここでは開発環境として MSYS2 を使用します。MSYS2 にツールチェーンとして [MinGW](#) をインストールします。

```
# pacman -S mingw-w64-i686-toolchain
# pacman -S mingw-w64-x86_64-toolchain
```


mingw-w64-*-toolchain は MinGW のパッケージグループです。32bit(i686) および 64bit(x86_64) の環境で動作するアプリケーションを作成するためのツールチェーンを個別にインストールします。MinGW を使用するにはスタートメニューから"MSYS2 MinGW 32-bit"もしくは"MSYS2 MinGW 64-bit"のいずれかを選択します。^{*21}

MinGW にはテキストエディタが含まれていないので別途用意します。MSYS2 には [Geany](#)/[gedit](#)/[Kate](#)/[Vim](#) などのテキストエディタが用意されています。文字コードとして Unicode が扱えるものであれば Windows 用のテキストエディタでも構いません。^{*22}

19.4 プログラミングの作業工程

19.4.1 Hello World

細かい理屈は後回しにして実際に簡単なプログラムを作って動かしてみましょう。文字化けを防ぐためテキストエディタでソースコードを保存する際は文字コードとして UTF-8 を指定してください。

以下のコードを記述したソースファイル hello.c を作成します。

↓ hello.c

```
#include <stdio.h>
```

```
int main(void) {  
    printf("hello, world!\n");  
    return 0;  
}
```

コマンド gcc を入力して hello.c をコンパイルし出力された a.exe を実行します。

```
# gcc hello.c  
# ./a.exe
```

hello, world!と表示されたら成功です。

^{*21} プログラムのコーディングには [Visual Studio Code](#) を使用する人が多いようです。

^{*22} MSYS2 MinGW 64-bit の環境で作成したプログラムは 32bit の環境では動作しないので注意してください。

19.4.2 メッセージボックスの表示

メッセージボックスを表示してみます。以下のコードを記述したソースファイル message.c を作成します。

↓ message.c

```
#include <windows.h>

int WINAPI WinMain(HINSTANCE hInst, HINSTANCE hPrevInst, LPSTR pCmdLine,
                  int showCmd) {
    MessageBox(NULL, "やあ、こんにちは", "メッセージ", MB_OK);
    return 0;
}
```

コマンド gcc を入力して message.c をコンパイルします。今回はオプション-o で出力する実行ファイルを指定しています。-fexec-charset=CP932 は文字化けを防ぐためのオプションです。Windows アプリケーションではテキストデータを Unicode(UTF-8) ではなく Shift-JIS(CP932) として処理しています。

```
# gcc message.c -o message.exe -fexec-charset=CP932
# ./message.exe
```

メッセージボックスが表示されたら成功です。

19.4.3 メインウィンドウの表示 (その 1)

ウィンドウを表示してみます。以下のコードを記述したソースファイル window.c を作成します。

↓ window.c

```
#include <windows.h>

// 記号定数
#define WINDOW_CLASS "foo" // ウィンドウクラス名
#define WINDOW_TITLE "サンプルウィンドウ" // タイトルバーの文字列
```

```

// プロトタイプ宣言
LRESULT CALLBACK WindowProc(HWND hWnd, UINT msg, WPARAM wp, LPARAM lp);

// エントリポイント
int WINAPI WinMain(HINSTANCE hInst, HINSTANCE hPrevInst, LPSTR lpCmdLine,
                  int iCmdShow) {
    WNDCLASSEX wc; // ウィンドウクラス構造体
    HWND hWnd;     // ウィンドウハンドル
    MSG msg;       // メッセージ構造体

    // ウィンドウクラスを設定
    wc.cbSize = sizeof(wc); // 構造体のサイズ
    wc.style = CS_HREDRAW | CS_VREDRAW; // スタイル
    wc.lpfnWndProc = WindowProc; // ウィンドウプロシージャ
    wc.cbClsExtra = wc.cbWndExtra = 0; // 拡張領域のサイズ
    wc.hInstance = hInst; // インスタンスハンドル
    wc.hIcon = LoadIcon(NULL, IDI_APPLICATION); // アイコン
    wc.hIconSm = wc.hIcon; // アイコン (小)
    wc.hCursor = LoadCursor(NULL, IDC_ARROW); // カーソル
    wc.hbrBackground = (HBRUSH)(COLOR_WINDOW); // 背景色
    wc.lpszMenuName = NULL; // メニュー名
    wc.lpszClassName = WINDOW_CLASS; // ウィンドウクラス名

    // ウィンドウクラスを登録
    if (RegisterClassEx(&wc) == 0) {
        return 1;
    }

    // ウィンドウを作成
    hWnd = CreateWindow(WINDOW_CLASS, // ウィンドウクラス名
                       WINDOW_TITLE, // タイトルバーの文字列
                       WS_OVERLAPPEDWINDOW, // スタイル
                       CW_USEDEFAULT, CW_USEDEFAULT, // 位置
                       CW_USEDEFAULT, CW_USEDEFAULT, // サイズ
                       NULL, // 親ウィンドウのハンドル

```

```

        NULL,    // メニューハンドル
        hInst,   // インスタンスハンドル
        NULL     // lp に渡す構造体のポインタ
    );
    if (hWnd == NULL) {
        return 1;
    }

    // ウィンドウを表示
    ShowWindow(hWnd, iCmdShow);
    UpdateWindow(hWnd);

    // メッセージループ
    while (TRUE) {
        int i = GetMessage(&msg, NULL, 0, 0);
        if (i == 0 || i == -1) {
            break;
        }
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }

    return 0;
}

// ウィンドウプロシージャ
LRESULT CALLBACK WindowProc(HWND hWnd, UINT msg, WPARAM wp, LPARAM lp) {
    switch (msg) {
        case WM_DESTROY:
            PostQuitMessage(0);
            break;
        default:
            break;
    }
    return DefWindowProc(hWnd, msg, wp, lp);
}

```

```
}
```

コマンド `gcc` を入力して `window.c` をコンパイルします。

```
# gcc window.c -o window.exe -fexec-charset=CP932
# ./window.exe
```

ウィンドウが表示されたら成功です。かなり長いコードですがこれが雛形です。処理の流れを簡単に説明します。

1. ウィンドウクラスを元にウィンドウを作成します。^{*23}
2. メッセージループを開始します。
3. イベントが発生するとシステムがメッセージを発行します。
4. メッセージループがメッセージを受け取りウィンドウプロシージャに送信します。
5. ウィンドウプロシージャがメッセージを受け取りイベントを処理します。^{*24}
6. アプリケーションが終了するまで3~5を繰り返します。

イベントの処理を記述することが Windows プログラミングの主な作業になります。このようなプログラムをイベントドリブン (イベント駆動型) といいます。ウィンドウプロシージャはシステムから自動的に呼び出されるという意味でコールバック関数とも呼ばれます。

19.4.4 メインウィンドウの表示 (その2)

メニューのついたウィンドウを表示してみます。以下のコードを記述したソースファイル `window.c` を作成します。

⇓ `window.c`

```
#include <windows.h>
```

```
// 記号定数
```

```
#define WINDOW_CLASS "foo" // ウィンドウクラス名
#define WINDOW_TITLE "サンプルウィンドウ" // タイトルバーの文字列
```

^{*23}

(a) ウィンドウクラスは C++ のクラスとは関係ありません。混同しないように注意してください。

^{*24} イベントについての情報はパラメータとしてウィンドウプロシージャに渡されます。

```

// プロトタイプ宣言
LRESULT CALLBACK WindowProc(HWND hWnd, UINT msg, WPARAM wp, LPARAM lp);

// エントリポイント
int WINAPI WinMain(HINSTANCE hInst, HINSTANCE hPrevInst, LPSTR lpCmdLine,
                  int iCmdShow) {
    WNDCLASSEX wc; // ウィンドウクラス構造体
    HWND hWnd;     // ウィンドウハンドル
    MSG msg;       // メッセージ構造体

    // ウィンドウクラスを設定
    wc.cbSize = sizeof(wc); // 構造体のサイズ
    wc.style = CS_HREDRAW | CS_VREDRAW; // スタイル
    wc.lpfnWndProc = WindowProc; // ウィンドウプロシージャ
    wc.cbClsExtra = wc.cbWndExtra = 0; // 拡張領域のサイズ
    wc.hInstance = hInst; // インスタンスハンドル
    wc.hIcon = LoadIcon(NULL, IDI_APPLICATION); // アイコン
    wc.hIconSm = wc.hIcon; // アイコン (小)
    wc.hCursor = LoadCursor(NULL, IDC_ARROW); // カーソル
    wc.hbrBackground = (HBRUSH)(COLOR_WINDOW); // 背景色
    wc.lpszMenuName = NULL; // メニュー名
    wc.lpszClassName = WINDOW_CLASS; // ウィンドウクラス名

    // ウィンドウクラスを登録
    if (RegisterClassEx(&wc) == 0) {
        return 1;
    }

    // ウィンドウを作成
    hWnd = CreateWindow(WINDOW_CLASS, // ウィンドウクラス名
                       WINDOW_TITLE, // タイトルバーの文字列
                       WS_OVERLAPPEDWINDOW, // スタイル
                       CW_USEDEFAULT, CW_USEDEFAULT, // 位置
                       CW_USEDEFAULT, CW_USEDEFAULT, // サイズ
                       NULL, // 親ウィンドウのハンドル

```

```

        NULL,    // メニューハンドル
        hInst,   // インスタンスハンドル
        NULL     // lp に渡す構造体のポインタ
    );
    if (hWnd == NULL) {
        return 1;
    }

    // ウィンドウを表示
    ShowWindow(hWnd, iCmdShow);
    UpdateWindow(hWnd);

    // メッセージループ
    while (TRUE) {
        int i = GetMessage(&msg, NULL, 0, 0);
        if (i == 0 || i == -1) {
            break;
        }
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }

    return 0;
}

// ウィンドウプロシージャ
LRESULT CALLBACK WindowProc(HWND hWnd, UINT msg, WPARAM wp, LPARAM lp) {
    static HMENU hMenu; // メニューハンドル
    switch (msg) {
        case WM_CREATE:
            hMenu = LoadMenu(NULL, "IDR_MENU");
            SetMenu(hWnd, hMenu);
            break;
        case WM_COMMAND:
            switch (LOWORD(wp)) {

```

```

        case 40001:
            DestroyWindow(hWnd);
            break;
        case 40002:
            MessageBox(NULL, "Version 1.00", "バージョン情報", MB_OK);
            break;
    }
    break;
case WM_CLOSE:
    SetMenu(hWnd, NULL);
    DestroyMenu(hMenu);
    hMenu = NULL;
    break;
case WM_DESTROY:
    PostQuitMessage(0);
    break;
default:
    break;
}
return DefWindowProc(hWnd, msg, wp, lp);
}

```

以下のコードを記述したリソースファイル resource.rc を作成します。メニューやダイアログなどのリソースデータはリソースファイルに記述します。

⇓ resource.rc

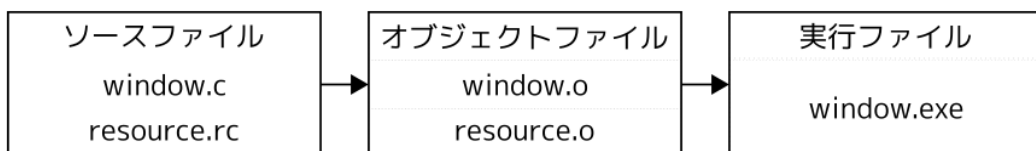
```

IDR_MENU MENU {
    POPUP "ファイル (&F)" {
        MENUITEM "終了 (&X)", 40001
        MENUITEM "バージョン情報 (&A)", 40002
    }
}

```

今まではソースファイルから直接実行ファイルを作成していましたが今回はリソースファイルが追加されたので次のようにします。リソースファイルのコンパイルにはリソースコンパイラ windres を使用します。

1. コマンド `gcc` を入力して `window.c` をコンパイルし `window.o` を生成
2. コマンド `windres` を入力して `resource.rc` をコンパイルし `resource.o` を生成
3. コマンド `gcc` を入力して `window.o` と `resource.o` をリンクし `window.exe` を生成



ソースファイルが複数ある場合はオブジェクトファイル (*.o) という中間ファイルを介して実行ファイルを作成します。-c オプションを指定するとオブジェクトファイルが生成されます。コマンド `gcc` には他にも様々なオプションを指定することができます。詳細については [gcc コマンドの使い方: UNIX/Linux の部屋](#) を参照してください。

```
# gcc -c window.c -fexec-charset=cp932
# windres -c 65001 resource.rc resource.o
# gcc window.o resource.o -o window.exe
# ./window.exe
```

メニュー付きのウィンドウが表示されたら成功です。

19.4.5 ソースファイルの分割

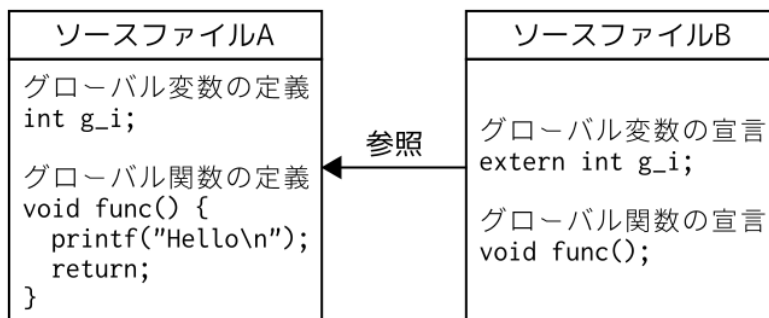
ソースコードは処理内容に応じて複数のファイルに記述するのが一般的です。ソースファイルを分割する際に注意すべきことはグローバルな変数・関数のリンケージです。リンケージには内部リンケージと外部リンケージがあります。

グローバルな変数・関数に `static` という修飾子をつければ内部リンケージになり何もつけなければ外部リンケージになります。ただし C++ ではグローバルな定数 (`const`) に何もつけなければ内部リンケージになるので注意してください。

内部リンケージの変数・関数はそれらが定義されたファイルからしか参照することができません。

外部リンケージの変数・関数はどのファイルからも参照することができます。外部リンケージの変数を参照するには参照元のファイルに `extern` という修飾子をつけて参照先の変数を宣言する必要があります。外部リンケージの関数を参照するには参照元のファイルに参照先の関数のプロトタイプ宣言が必要です。当然ですが外部リンケージの変数・関数の定義が複数のソー

スファイルで重複することは許されません。また内部リンケージと外部リンケージの識別子が重複することも許されません。



19.4.6 ビルド作業の自動化

上のようにコマンドを入力して実行ファイルを作成する作業をビルドといいます。詳細については[ゼロから学ぶ C++](#)を参照してください。

コマンド make を使用すればビルドを自動化することができます。以下の内容を記述した Makefile(もしくは makefile) というファイルを作成してください。インデントにはタブを使用します。スペースでインデントするとエラーになるので注意してください。

↓ Makefile

```
TARGET = window.exe # 生成される実行ファイル
SRCS = $(notdir $(shell find ./ -name '*.c')) # コンパイルするソースファイル (*.c)
OBS = $(SRCS:.c=.o) resource.o # コンパイルするオブジェクトファイル (*.o)
CC = gcc # 使用するコンパイラ
CFLAGS = \ # コンパイラのオプション
    -s -O2 -Wall -fexec-charset=cp932 # リリース用
# -g -O0 -Wall -fexec-charset=cp932 # デバッグ用

.PHONY: all clean

all: $(TARGET)

# オブジェクトファイルから実行ファイルを作成
```

```
$(TARGET): $(OBJS)
    $(CC) -o $@ $^

# ソースファイルからオブジェクトファイルを作成
%.o: %.c
    $(CC) -o $@ $(CFLAGS) -c $<

# リソースファイルからオブジェクトファイルを作成
%.o: %.rc
    windres -c 65001 -i $< -o $@

# オブジェクトファイルを削除
clean:
    -@rm -f $(OBJS)
```

コマンド make を入力すると Makefile の内容に従って実行ファイル window.exe が作成されます。make clean と入力すればオブジェクトファイルが削除されます。詳細については[自動化のための GNU Make 入門講座/コマンド:make: UNIX/Linux の部屋/make コマンド - IBM Documentation/GNU make 日本語訳](#)などを参照してください。

19.4.7 メインウィンドウの表示 (その3)

wxWidgets を使用してウィンドウを表示してみます。

MSYS2 で wxWidgets をインストールします。

```
# pacman -S mingw-w64-i686-wxmsw3.1
# pacman -S mingw-w64-x86_64-wxmsw3.1
```

以下のコードを記述したソースファイル window.cpp を作成します。

```
↓ window.cpp

#include <wx/wx.h>

// アプリケーションの単一クラス
class MyApp : public wxApp {
public:
```

```

    virtual bool OnInit();
};

// メインウィンドウのフレームクラス
class MyFrame : public wxFrame {
public:
    MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size);

private:
    void OnExit(wxCommandEvent& event);
    void OnAbout(wxCommandEvent& event);
};

// アプリケーションを初期化
bool MyApp::OnInit() {
    MyFrame* pFrame =
        new MyFrame(_("サンプルウィンドウ"), wxDefaultPosition, wxSize(640, 480));
    pFrame->Show(true);
    this->SetTopWindow(pFrame);
    return true;
}

// メインウィンドウを初期化
MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size)
    : wxFrame(NULL, -1, title, pos, size) {
    wxMenu* pMenu = new wxMenu;
    pMenu->Append(wxID_ABOUT, _("バージョン情報 (&A)"));
    pMenu->Append(wxID_EXIT, _("終了 (&X)"));

    wxMenuBar* pMenuBar = new wxMenuBar;
    pMenuBar->Append(pMenu, _("ファイル (&F)"));
    SetMenuBar(pMenuBar);

    this->CreateStatusBar();
    Center();
}

```

```

// イベントとハンドラを結合
Bind(wxEVT_MENU, &MyFrame::OnAbout, this, wxID_ABOUT);
Bind(wxEVT_MENU, &MyFrame::OnExit, this, wxID_EXIT);
}

// OnExit ハンドラ
void MyFrame::OnExit(wxCommandEvent& WXUNUSED(event)) { Close(true); }

// OnAbout ハンドラ
void MyFrame::OnAbout(wxCommandEvent& WXUNUSED(event)) {
    wxMessageBox(_("Version 1.00"), _("バージョン情報"),
        wxOK | wxICON_INFORMATION, this);
}

IMPLEMENT_APP(MyApp)

```

コマンド `g++` を入力して `window.cpp` をコンパイルします。

```

# g++ window.cpp -o window -fexec-charset=cp932 'wx-config-3.1 --cxxflags --libs'
# ./window.exe

```

ウィンドウが表示されたら成功です。WindowsAPI を使用した場合と比較してソースコードがかなり短くなりました。

`wxWidgets` は C++ で使用できる GUI ツールキット (C++ のクラスライブラリ) です。`wxWidgets` には便利で高機能なクラスがたくさん用意されています。[wxFormBuilder](#) などの GUI ビルダと併用すれば効率的に GUI のアプリケーションを作成することができます。詳細については [wxWidgets 日本語ドキュメントプロジェクト](#) を参照してください。

19.5 プログラミングの学習方法

MSYS2 はプログラミングの学習に最適な環境です。MSYS2 でサンプルコードを繰り返し実行する方法を紹介します。

`a.exe` を簡単に実行できるようにするためあらかじめ以下のようなエイリアスを設定しておきます。

```
alias foo='$HOME/a.exe'
```

以下のようにコマンド cat を入力してコピーしたサンプルコードを貼り付け Ctrl-d を入力します。

```
# cat >foo.c
```

コマンド gcc を入力してサンプルコードをコンパイルし実行します。

```
# gcc foo.c
```

```
# foo
```

プログラミングは習うより慣れろです。見よう見まねでやっていけば自然に身につきます。覚えることはたくさんありますが単純なことを 1 つ 1 つ積み重ねていくだけです。そして理屈よりも実際にプログラムを動かしてプログラミングに親しむことが大切です。