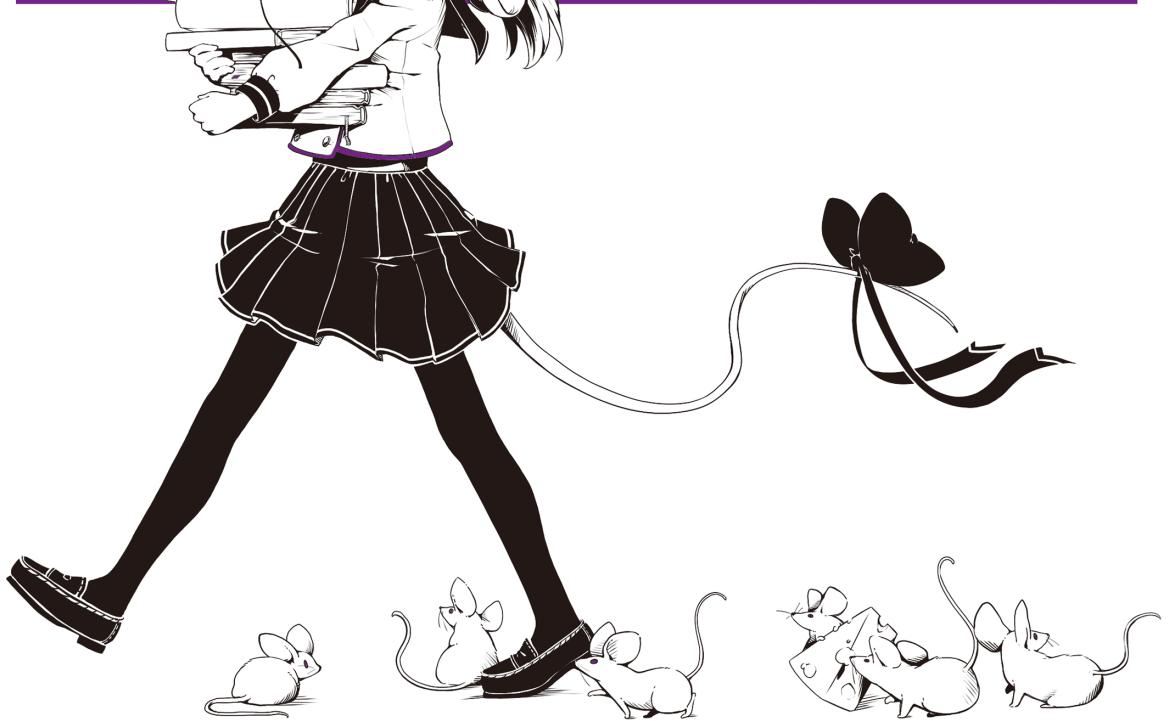


Let's enjoy Coreutils life!

第7版

解説 Core Utils



DT
第7開発セクション

tboffice 著
矢上 謙一 画

解説CoreUtils

第7版

tboffice著
第7開発セクション

はじめに

1.1 まえがき

「情報というのは面白いもので、こちらから探しにいかないとないのと同じで、ゲームなどでも攻略に行き詰まつたりして、ネットで Wiki を調べてみたり、攻略本を探してようやく出てくるという性質でして、人間、必要に駆られないと、なかなか新しい情報を得ようとしない^{*1} ものです。」^{*2}

1.2 この本は何？

この本は、GNU Coreutils^{*3} の執筆時点の最新のマニュアル^{*4} をひと通り眺めてみて、解説をしてみようというコンセプトの本に書かれた本です^{*5}。Coreutils 8.31 (2019-03-10 release) の内容まで盛り込んでいます。

^{*1} なかなか新しい情報を得ようとしない：新しい情報が得られる便利な仕組みはいくつもあるものの、そういうツールが増えれば増えるほど「自分にとって都合のよい新しい情報」しか目に入らなくなるのが人間の愛おしさである。

^{*2} 蟬丸 P (2012). 蟬丸 P のつれづれ仏教講座 株式会社エンタープライズ p.263

^{*3} <http://www.gnu.org/software/coreutils/>

^{*4} <http://www.gnu.org/software/coreutils/manual/>

^{*5} もともとは、弊サークル・第 7 開発セクションが発行してきた同人誌「ななか Inside PRESS」の 2 号と 3 号にまたがって掲載された「Coreutils 大全」という連載を一冊の本にまとめたものが初版でした

1.3 対象読者

Linux のコンソールでお仕事をしていると、シェルスクリプトで楽をしようとなります。楽をするため、筆者がよく使うコマンドを調べていたところ、Coreutils のマニュアルにたどり着きました。気づけば、Coreutils のマニュアルを読んで、コマンドの実行方法や、知らなかったコマンドや、今まで使ったことがないオプションを見つけていました。

たとえば、

- ・「cat をたんごくでじっこうしたら どうなるです？」
- ・「ふあいるを pr | pr したらどうなるです？」「べとべとです？」
- ・「tail -f と tailf はどこがちがうです？」
- ・「だーくふれいむますたー！」(指を指して)「あのころのきおくをけすにはどうやるです？」
- ・「ばいなりでーたおいしいです」 *6
- ・「イエッス！アスマス！」
- ・「ここはどこ？わたしはだあれ？」
- ・「せんせい！そいんすうぶんかいがしたいです！」 *7

普段 linux を使っているけど、このコマンド知らなかった、こんな使い方ができるかのという発見が一つでもあれば幸いです。筆者がよく使うコマンドや Tips、実行例も載せています。それでは、Coreutils の世界へようこそ。

1.4 ご注意

基本的に、ソースを読んで実装部分などの話はありません。コマンド・オプションの使い方を説明しています。バージョンにより、使えないコマンドやオプションがあります。本書はバージョン 8.27 のマニュアルを元にコマンドの解説を行っています。また、筆者が検証した環境は、DigitalOcean(CentOS 6.6) *8 と amazon EC2 *9 のマイクロインスタンスです。パッケージはほぼデフォルトのため、CentOS 6.6 では Coreutils 8.4、amazon EC2 では 8.21 です。本文中でてくるコマンドのサンプルは、上記の環

*6 どこぞの妖精さんをお借りしました。扱いにくいキャラクターですね。文章にするとかどくせいがひじょうにわるいです？

*7 同僚の@saiten 氏に描いていただきました。妖精さんの画像を要請したらこんなイラストが上がってきました。イラスト技術が養成されたんじゃないでしょうか（ドヤア

*8 CentOS 6.5 のイメージがあり、yum update すると 6.6 になった

*9 /etc/issue 曰く、Amazon Linux AMI release 2014.03



Figure1.1 働く妖精 (?) さん

境で実行していますが、使うときは誰にも迷惑をかけないところで実験・検証してください。

この本では、Coreutils のマニュアルの冗長な部分はできるだけ端折って、面白そうなところや、実践的な部分を取り上げました。もし時間があるようであれば、Coreutils のマニュアルを上から下まで読んでみてください^{*10}。新たな発見があると思います。また、文章の都合上、Coreutils にないコマンドも載っています^{*11}。

本文中の「原文」は Coreutils のマニュアルのことを指します。

1.5 そもそも **Coreutils** とは

「コアユーティルズ」と読みます。文字に落としてみるとダサいですね。まあ、そのへんは置いといて、ls や cat など、linux では欠かせないコマンドをまとめたパッケージです^{*12}。Fileutils, Shellutils, Textutils という utils を統合したものです。

^{*10} 正直なところ、かなり面倒です。あまり面白くないです

^{*11} 怒らないよね？おこなの？(やめろってば

^{*12} この本をお手にとっている人に、Coreutils の説明をしてもあまり意味がないような気もしますが、まあ、一応

ChangeLog をみたところ、一番古い日付は 2002-07-01 でした。おそらくそのころに統合されたのでしょう。メジャーバージョンは 2003 年 4 月にバージョン 5 として登場しました。2019 年 3 月現在、8.31 (2019-03-10) がリリースされています^{*13}。

Mac にも Coreutils と同じく、ls や cat などのコマンドがあります。こちらは BSD 由来なので、GNU Coreutils の実装と異なっています。Mac に GNU Coreutils をインストールしたいときは、homebrew でインストールできます。

```
brew install coreutils
```

なお、既存のコマンドと名前がかぶるので、プレフィックスに g がついています。ls だったら gls となっています。od だった場合は・・・もうあとは分かるな・・・？

1.6 本稿の構成

最初に Coreutils パッケージのコマンドに共通のオプションを解説し、Coreutils のマニュアルの通りにコマンドをならべて解説しています。一部、冗長なところがあるのでまとめたりしています。問題ないでしょう。

1.7 そんなことより

ソースどこだよ：github にあります <https://github.com/coreutils/coreutils/>
zip でくれ：<http://ftp.gnu.org/gnu/coreutils/> か <https://github.com/coreutils/coreutils/releases>
頻繁に寄せられる質問は：FAQ です <http://www.gnu.org/software/coreutils/faq/coreutils-faq.html>
マニュアルは：<http://www.gnu.org/software/coreutils/manual/>
メーリングリストは：<http://lists.gnu.org/archive/html/coreutils/>

^{*13} 現在でも数日おきにバグフィックスなどが入っています

Contents

第 1 章	イントロダクション	1
第 2 章	よくあるオプション	2
2.1	--help	2
2.2	--version	3
2.3	--	3
2.4	-	3
2.5	その他、Coreutils の共通したこと	4
第 3 章	この素晴らしいファイルに出力を!	5
3.1	cat	5
3.2	tac	6
3.3	nl	7
3.4	od	8
3.5	base64	8
3.6	base32	8
3.7	basenc	9
第 4 章	てーさい	10
4.1	fmt	10
4.2	pr	10
4.3	fold	11
第 5 章	ファイルの一部を出力	12
5.1	head	12
5.2	tail	12
5.3	split	13

5.4 csplit	14
第 6 章 ファイルの要約系	16
6.1 wc	16
6.2 sum	16
6.3 cksum	17
6.4 b2sum	17
6.5 md5sum	18
6.6 sha 系	18
第 7 章 かぐや様は仕分けたい	19
7.1 sort	19
7.2 shuf	21
7.3 uniq	23
7.4 comm	23
7.5 ptx	24
7.6 tsort	24
第 8 章 テーブルの欄操作	26
8.1 cut	26
8.2 paste	27
8.3 join	27
第 9 章 キャラクタ操作	29
9.1 tr	29
9.2 expand	29
9.3 unexpand	30
第 10 章 ファイルリスト表示	31
10.1 ls	31
どんなファイルを表示するか	31
どんな情報を表示するか	32
ソート順を指定	33
表示のフォーマットを指定	33
タイムスタンプの表示形式	34
ファイル名の表示形式	34

10.2	dir	35
10.3	vdir	35
10.4	dircolors	35
第 11 章	基本的操作	36
11.1	cp	36
11.2	dd	38
11.3	install	38
11.4	mv	39
11.5	rm	39
11.6	shred	41
第 12 章	スペシャルファイルタイプ	42
12.1	link	42
12.2	ln	42
12.3	mkdir	43
12.4	mkfifo	43
12.5	mknod	44
12.6	readlink	44
12.7	rmdir	44
12.8	unlink	45
第 13 章	ファイルの属性を変更	46
13.1	chown	46
13.2	chgrp	46
13.3	chmod	47
13.4	touch	47
第 14 章	このディスク容量には問題がある!	48
14.1	df	48
14.2	du	49
14.3	stat	49
14.4	sync	50
14.5	truncate	50
第 15 章	文字を表示	51

15.1 echo	51
15.2 printf	51
15.3 yes	51
第 16 章 条件	53
16.1 false	53
16.2 true	53
16.3 test	55
16.4 [.	56
16.5 expr	56
第 17 章 リダイレクション	57
17.1 tee	57
第 18 章 ファイル名の操作	58
18.1 basename	58
18.2 dirname	58
18.3 pathchk	59
18.4 mktemp	59
18.5 realpath	60
第 19 章 わーきんぐの状況	61
19.1 pwd	61
19.2 stty	62
19.3 printenv	63
19.4 tty	63
第 20 章 ユーザの情報	64
20.1 id	64
20.2 logname	65
20.3 whoami	65
20.4 groups	65
20.5 users	66
20.6 who	66
20.7 pinky	67

第 21 章	システムコンテキスト	69
21.1	date	69
21.2	arch	71
21.3	nproc	72
21.4	uname	72
21.5	hostname	73
21.6	hostid	73
21.7	uptime	73
第 22 章	SELinux	75
22.1	chcon	75
22.2	runcon	75
第 23 章	変更コマンド	77
23.1	chroot	77
23.2	env	77
23.3	nice	78
23.4	nohup	79
23.5	stdbuf	80
23.6	timeout	81
第 24 章	プロセスコントロール	83
24.1	kill	83
第 25 章	遅延	85
25.1	sleep	85
第 26 章	数値操作	86
26.1	factor	86
26.2	numfmt	87
26.3	seq	88
第 27 章	答え合わせ	90
27.1	date	90
27.2	factor	90
第 28 章	おわりに	92

28.1 第 2 版おわりに	92
28.2 第 3 版おわりに	93
28.3 第 4 版おわりに	93
28.4 第 5 版おわりに	93
28.5 第 6 版おわりに	94
28.6 第 7 版おわりに	95
索引	96

第 1 章

イントロダクション

ここから、Coreutils のマニュアルに入ります。Coreutils のマニュアルの第 1 章は、「Introduction」から始まっています。少し読んでみましょう。

Coreutils のマニュアルは、絶賛製作中です！初心者向けに基本的な概念は説明してないよ！興味があれば改善していってね。あと、バグを見つけたら bug-coreutils@gnu.org に送ってね。そのとき、再現環境も一緒に送ってね！^{*1} だそうです。

^{*1} あとは先人たちの名前が書かれています

第 2 章

よくあるオプション

例えば、`ls -l` の`-l`がオプションです。ちなみに、`sort -r password -t :`と実行すると、`sort -r -t : password`として実行したように動作します。コマンドによっては、ものすごい長いオプション名がありますが、省略することができます。`rmdir --ignore-fail-on-non-empty`の場合であれば、`rmdir --ignore-fail`とか`rmdir --i`で実行できます。`ls --h`のときは、複数のオプションにマッチするので「分からへん」^{*2}と言われたり、「どれやねん」^{*3}と言われたりします。

以下、Coreutils での共通のオプションです。

2.1 --help

たとえば`ls`のコマンドのオプションを調べたいとき

```
$ ls --help
```

簡易ヘルプはこのオプションをつかえば問題☆解決。え？問題が解決しない？そんなときはこのようなコマンドを打ってください。

```
$ man ls  
$ info coreutils ls
```

`info` コマンドを打了後は、Emacs のキーバインドなのでそこさえ気をつけければ`info` コマンドは良いコマンドだと思いますよ（白目

^{*2} (coreutils 8.4) `ls`: option ‘-h’ is ambiguous

^{*3} (coreutils 8.25) `ls`: option ‘-h’ is ambiguous; possibilities: ‘-human-readable’ ‘-help’ ‘-hide’ ‘-hide-control-chars’

2.2 --version

バージョンを表示します。

2.3 --

このオプションを打った後は、オプションが無効になります。たとえば、こんなコマンドを打ったとき

```
$ ls -l -- -l
```

-l というファイルあるいはディレクトリを ls -l で表示します。

```
$ mkdir -- -l
```

とやると、 -l というディレクトリが出来ます^{*4}。touch でも同様。

2.4 -

ただのハイフンです。

オプションじゃないよ！^{*5}。オプションに見えるだけだよ？ホントだよ？すたんだーどいんぶつとを待つ記号だよ！受け取った結果はコマンドがよろしく処理するよ！たとえばこんな感じだよ！あとで出てくるから覚えておいてね！

```
$ sort -
b # 打つべし
a # 打つべし
c # 打つべし
Ctrl-d # ctrl 押しながら d って打つと下記のように表示されるよ !
a
b
c
```

^{*4} そんな名前のディレクトリとかファイル作るなよ！あとで面倒だぞ！！改行コードだけのファイル名も作れるぞ！作るなよ！！（フラグ

^{*5} と、マニュアルに書いてあります

2.5 その他、Coreutils の共通したこと

2章は、2.1章から2.14章まであります。かいつまんで、書いてあることを説明します。

返り値 (Exit Status) コマンドを実行したときに数値が返ります。コマンドを実行したあとにすぐ echo \$? をやると出てくる数値です。0 が通常にコマンドが終わったことを示し、1 は異常があったことを示します。0,1 以外の数値を返すコマンドもあります。chroot, env, expr, nice, nohup, numfmt, printenv, sort, stdbuf, test, timeout, tty です。

バックアップオプション -b オプションです。cp, install, ln, mv にあります。ファイル操作するときに元のファイルをどのようにバックアップするか指定します。サフィックスをつけるなら -S オプションです。詳細は各コマンドを参照してください。

ブロックサイズ blocks のサイズを設定することができます。キロバイト (kB) やキビバイト (K,k,KiB) とかのあれです。df, du, lsあたりで使います。

ユーザ名と ID の曖昧さの除去 ユーザ名が数字の場合どうなってしまうんでしょうか。そのあたりは、chownあたりのコマンドに詳細を書きました。

ランダムデータのソース sort -R コマンドで --random-source=file を与えると file を元にランダムにソートします。詳細は shuf コマンドを参照してください。

スペシャルビルトインコマンド

```
  nice . foo.sh
  nice :
  nice exec pwd
```

といったコマンドは意図した結果にはなりません。bash にもスペシャルビルトインコマンドがあって、nice suspend とかできません。ビルトインコマンドには、下記があります。

```
. : break continue eval exit export readonly return set shift
times trap unset
```

他には、浮動小数や、シグナル一覧や、ディレクトリの指定の方法や、/の取り扱い方、symlink のたどり方などがあります。

第3章

この素晴らしいファイルに出力を！

すごい高貴なファイルの気配がありますね。え？なさそう？そう言われるとそんな気もする^{*6}。

3.1 cat

Linux の基礎として間違いなく出てくるコマンドではないでしょうか。ファイルを引数にとると、そのファイルの中身を表示するコマンドです^{*8}。`file` というファイルの中身を出力するときにはこうします^{*7}。

```
$ cat file
```

次に、マニュアルにあるオプションの読み方を説明します。説明しますよ！！！^{*9} マニュアルには

```
cat [option] [file]...
```

とあります^{*10}。

- [] この括弧で囲まれている部分は省略可能です
- ... この記号は引数がいくつでもいいよって言ってます
- つまりこんな感じでコマンドを作ることが出来ます。

^{*6} 原題は、Output of entire files。元ネタは、「この素晴らしい世界に祝福を！」。通称このすば。ライトノベル原作、異世界転生モノ。アニメ化され、作画崩壊を思わせる微妙な作画崩壊が癖になる

^{*8} 説明が足りないと気づいた方は銳いです。coreutils のマニュアルの通りの説明は後ほど

^{*7} でかいファイルとかバイナリとか食わせるなよーどうなっても知らんぞ！！持ち主の分からぬファイルは `ls -lh` とか `file` コマンドで確認するんだぞ！！

^{*9} 重要なことなので2回書きました

^{*10} ほかのコマンドのマニュアルもほぼ同様に書かれています

3.2. tac

```
$ cat -A -n hogefile fugafile piyofile
```

主なオプションの説明をします。

-E	行末がどこまで入ってるか分からないから表示して
-n	行数を付けて
-T	タブ文字も表示して欲しいなあ
-v	改行文字も表示して欲しいなー
-A	-vETと同じ。改行文字、行末文字、タブも表示します
-s	連続する空行をひとつにまとめる

オプションの FILE 部分に - を入れると標準入力になります。f と g というファイルがあって、

```
cat f - g
```

このようにコマンドが実行されたときは、f の内容を表示、標準入力の内容表示、ctrl-d(ctrl を押しながら d を押すこと/意味は、入力の終わり)を押したら g の内容が表示されます。cat 単独で打ったときは — マニュアル通りの説明をすると、「ファイルまたは標準入力を標準出力にコピーする」 *11 です。引数が指定されていないときは標準入力になります。標準入力と標準出力を体感してください *12。

ちょっとしたテキストファイルを作るときは

```
$ cat > hogefile
hoge
fuga
piyo
# ここで ctrl+d を押す
```

と打つとよいです。

3.2 tac

さて、cat のあと tac。お察しの通りです。早速、実行してみましょう。ファイルを作るのが面倒なので echo します。

*11 cat copies each file (‘-’ means standard input), or standard input if none are given, to standard output.

*12 実際に打ってみよう!! ctrl + c で抜けられるよ!!

```
$ echo -e "a\nb\nc" | tac
c
b
a
```

はい、ファイルの内容を上下逆に出します。1行分の文字の並びを逆にするには、revコマンドを使ってください。誰得魔方陣の例。

```
$ echo -e "2 9 4\n7 5 3\n6 1 8" | tac | rev
```

単語単位で逆にするには、

```
$ tac -r -s '[^a-zA-Z0-9\-\-]' file
```

revと同じ効果をtacでやるには下記^{*13}。ただし日本語の2バイト文字列も逆にするので化けます。おちゃめさんですね（何

```
$ tac -r -s '.\\/' myfile
```

3.3 nl

行番号を振ってくれるコマンド。単に実行した場合はこんな感じです。

```
$ nl /etc/issue
1      CentOS release 6.6 (Final)
2      Kernel \r on an \m
```

デフォルトだと、空行には番号が付きません。なお、cat -b fileと同じです。オプションに-b aを付けると空行でも行番号がつきます。いろいろオプションがあるので値を変更してください。

```
$ echo -e "hoge\n\nfuga\npiyo" | nl -b a -n rz -s " hoge: " -v 3 -w 3
003 hoge:
004 hoge:
005 hoge: fuga
```

^{*13} 単語単位で逆にする例、とrevの効果の例は下記に載っていました。<http://bit.ly/SwZTt4>

3.4. od

```
006 hoge: piyo
```

3.4 od

主にバイナリファイルを8進数や16進数などで表示するコマンド。デフォルトでは8進数で表示。任意のバイト数をスキップしてそこから表示開始もできます。なんとなくxxdやhexdumpを使ってしまって、出番のないコマンドのような…

```
$ od /etc/issue
0000000 062503 072156 051517 071040 066145 060545 062563 033040
0000020 033056 024040 064506 060556 024554 045412 071145 062556
0000040 020154 071134 067440 020156 067141 056040 005155 000012
0000057
```

3.5 base64

データを印刷できる形式に変換するコマンド、とマニュアルには書いてあります。RFC 4648^{*14}に則ってデータを変換するコマンドで、133%データが大きくなります。デコードも出来ます。

```
$ base64 /etc/issue | base64 --decode -i
CentOS release 6.6 (Final)
Kernel \r on an \m
```

3.6 base32

Coreutils 8.25 から登場。RFC 4648 の Base32 を実装したものです^{*15}。デコードも出来ます。

```
$ base32 /etc/redhat-release | base32 --decode
CentOS Linux release 7.2.1511 (Core)
```

^{*14} <http://tools.ietf.org/html/rfc4648>

^{*15} 経緯：https://bugzilla.redhat.com/show_bug.cgi?id=1250113

3.7 basenc

この basenc^{*16} は Coreutils 8.31 から登場。様々なエンコーディングがこのコマンド一つでできるようになりました。使い方は：

```
basenc encoding [option]… [file]
basenc encoding --decode [option]… [file]
```

encoding は必須です。encoding を下記に示します。といってもハイフン 2 回から始まります。

--base64	base64 にエンコードする。-d または--decode オプションとともに使うとデコードする。base64 コマンドと同等
--base64url	file-and-url-safe base64 エンコードする。+ と/の代わりに_- が使われる。-d または--decode オプションとともに使うとデコードする
--base32	base64 と同様。base32 コマンドと同等
--base32hex	Hex Alphabet base32 にエンコードする。-d または--decode オプションとともに使うとデコードする。エンコードされたデータは ‘0123456789ABCDEFHIJKLMNOPQRSTUVWXYZ=’ の文字が使われる
--base64	base16(hexadecimal) にエンコードする。-d または--decode オプションとともに使うとデコードする。エンコードされたデータは ‘0123456789ABCDEF=’ の文字が使われる
--base2lsbf	バイナリ文字列 (0 と 1) にエンコードする。各バイトの最下位ビットが最初になる
--base2lsbf	バイナリ文字列 (0 と 1) にエンコードする。各バイトの最上位ビットが最初になる
--z85	Z85(修正 Ascii85) にエンコードする。-d または--decode オプションとともに使うとデコードする。エンコードされたデータには ‘0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ.-:+=?&<>()[]{}@%\$#’ 文字が使われる。-z85 でエンコードするときは入力の長さを 4 の倍数にしなければならない。-z85 でデコードするときは、5 の倍数でなければならない

^{*16} base encoding 由来らしい。詳細は Coreutils の ML 参照

第 4 章

てーさい

体裁を整えるコマンドたちです^{*17}。

4.1 fmt

テキストファイルの文字を適当に折り返してくれるコマンドです。デフォルトだと 75 文字で折り返します。すでに改行されてしまっているテキストファイルでもなんとかしてくれます^{*18}。

4.2 pr

ページ単位で印刷用に、ヘッダとフッタを自動的に追加してくれるコマンドです。RFC みたいな文章がすぐに出来る！プレーンテキストすばらしい！と筆者の脳内で大好評のコマンドです。お試しあれ。ただし使いどころは限定期です。なお、pr | pr するとヘッダとフッタが二重につくだけです。こんな感じです。

```
cat /etc/issue | pr | pr | head -n 12
```

2015-08-01 10:07

1 ページ

^{*17} 原題は、Formatting file contents。「てーさい」の元ネタは、てきゅう（第 9 期）です。「ていさい!!」にしても良かったかも。ぶおんぶおん!!フルスロットルで楽器をかきならせ!!いろいろ混ざってよくわかんなくなってきた

^{*18} wikipedia のサンプルが易しいです。<http://en.wikipedia.org/wiki/Fmt>

2015-08-01 10:07

1 ページ

```
CentOS release 6.6 (Final)
Kernel \r on an \m
```

4.3 fold

テキストファイルをぴったりの文字数で改行するコマンド。fmt は空気を読んで、単語をぶつたぎらないようにしていますが、このコマンドは空気を読まずにぶった切れます。fmt の様な挙動をさせるには、-s をつけるとある程度空気を読んでくれます。fmt は引用符の中は改行しませんが、この fold は改行します。

第 5 章

ファイルの一部を出力

5.1 head

ファイル名を引数に取ると、ファイルの最初の 10 行を表示するコマンドです。-n 5 で先頭 5 行を表示。-c 10KB で先頭 10 キロバイトを表示。バイナリファイルでも OK *²⁰ *¹⁹。-n のあとにマイナス値を打つとどうなるでしょうか。環境にもよりますが…自分でやってみてください。

5.2 tail

ファイル名を引数にとると、ファイルの最後の 10 行を表示するコマンドです。サーバ管理者は毎日打っていると言っても過言ではないです。-f オプションをつけることによって、引数にとったファイルに対して追加された文字が出てきます。ちなみに複数のファイルを食わせることができるので、アクセスログファイルとアクセスエラーログファイルの両方を tail -f で表示することも可能。パイプでつないで特定の文字列だけ出力することも可能。

```
tail -f access.log error.log | grep --color -E "(==/192.168)"
```

ログファイルから”==” または、”192.168” という文字列を抜き出しています *²¹。”==” というのは、やってみてのお楽しみ。

tail -f コマンドと同じような働きをする tailf コマンドがあります。結論から言うと、最新の Coreutils を使っているならどっちも変わりありません *²²。どちらも inotify イベントを受け

*²⁰ 標準出力に出力して、端末が化けても知らないですよ

*¹⁹ 宗教上の理由で head コマンドを打ちたくない人は、sed 10q と打ってください。

*²¹ ちなみに grep の--color オプションはこの URL で知りました。 <http://aerith.mydns.jp/regrets/2008/12/tail-color.html>

*²² coreutils version 7.5 で inotify に対応した模様です。ここを参照しました。
<http://dev.ariel-networks.com/Members/inoue/tailf/>

取って処理するようになっています。もしも、対象のファイルが消えてしまう、あるいは同じファイル名なんだけどログローテートして中身がリセットされるときは、ファイルを読み直す下記のオプションを使いましょう^{*23}。ついでに 8.24 から-f しているファイルがリネームされても内容を追っかけることが可能になりました。

```
$ tail -F filename
```

`tail -n +2 file` とすると、file ファイルの 1 行目だけ削れます。`mysql` コマンドに-e で SQL を打ち込むと一行目にカラム名でてくるじゃないですか、あれを削るときに使うんですよ^{*24}。あと、`less` でファイルを開いている時に、おもむろに F を押すと `tail -f` モードに切り替わります。ついでに、FreeBSD 系には、-r というオプションがあります。Coreutils では、`tac` コマンドで代用してください。

5.3 split

ファイルを分割するコマンドです。書式は下記です。

```
split [option] [input [prefix]]
```

デフォルトで実行するとこんな感じになります。

```
$ split hogefile
$ ls
hogefile xaa  xab  xac  xad  xae  xaf  xag  xah  xai
```

1000 行ごとに 1 ファイルを、カレントディレクトリに生成します^{*25}。`xaa xab ...` となっているのは、あとで `cat` すると元に戻る^{*26} からです。100 行ごとに分割してほしいとか、`xx` というファイル名いやだというときはこんな感じです。

```
$ split -l 100 hogefile AA
$ ls
```

^{*23} `follow` の f らしい。`--max-unchanged-stats=N` というオプションがあって、-F オプションを使った時に、何秒おきにファイルを見に行くか指定することができます。デフォルトは 5 秒です。一瞬だけ作成されるファイルの中身を書き出しておくときに使えるかも？

^{*24} おい、やけに具体的だな

^{*25} でかいサイズのファイルのときには注意。たくさんファイルができるよ！！

^{*26} `cat x*` する。x の次は y,z と使っていく。最後どうなるのか実験だ！

5.4. csplit

```
hogefile AAaa  AAab  AAac  AAad  AAae  AAaf
```

-b オプションで任意のバイト数で split することができます。分割しながら圧縮できる (filter に通す) というオプションもあります *²⁷ *²⁸ *²⁹ *³⁰ *³¹。

使いどころが非常に謎ですが、-n オプションの例を示します *³²。

```
$ seq 100 > k; split -nl/7/33 k
20
21
22
```

8.24 から指定した文字でファイルを分離することができるようになりました (-t オプション)。ASCII の NULL 文字でファイルを分割したいというときは split -t '\0'。

5.4 csplit

「ファイルを文脈ベースで分割する」コマンドです。端的には、特定の文字が出てきたら split するコマンドです。使いどころによっては非常に強力なコマンドです。書式は下記。

```
csplit [option]... input pattern...
```

下記のようにすると、xx00 に文字列を出力し、hoge という文字と遭遇したら、別のファイル(xx01)をつくり、そこに出力します。もとのファイルはそのまま残っています。xx01 のファイル名の 1 行目に hoge という文字が含まれています *³³。

```
$ csplit hogefile /hoge/
```

さてマニュアルを追ってみましょう。「ファイルがたくさんできるから、最初にディレクトリを作り、その中に cd しましょう」と書いてあります。やっておきましょう。

*²⁷ xz -dc BIG.xz | split -b200G -filter='xz > \$FILE.xz' - big- (マニュアルより。big-aa.xz, big-ab.xz といったようにファイルが出来上がります)

*²⁸ ディスクの単価が安い現代に需要があるかどうか...

*²⁹ あるって!開発環境とかいつもディスク枯渇してるじゃん!!

*³⁰ 開発環境で split する用途があるか疑問だにゃあ

*³¹ 脚注で会話するなよ

*³² [練習問題] 何をしているのか、マニュアルを読んで確認してみましょう

*³³ 長い文章をすっぽり二つに分割する時に便利。日本語文字列でも split できる。hoge 文字列からの offset が使えるのがさらに便利

```
$ mkdir d && cd d
```

次に 0 または 5 で終わる文字にマッチしたら、そこでまた別のファイルを作ってそこに出力します。 { *} があるので、マッチしたぶんだけファイルが生成されます。出力されている数字は、それぞれのファイルのバイト数です。

```
$ seq 14 | csplit - '/[05]$/' '{*}'  
8  
10  
15  
$ ls  
xx00  xx01  xx02
```

ファイルの中身が、なんとなくどうなっているか分かったところで終わりです。

第 6 章

ファイルの要約系

6.1 wc

ファイルの行数を知るときによく出るコマンドです。 `wc -l` が有名すぎて、`wc` 単体の結果については `man` を引かないと忘れてることが多いです。`wc file` したときは、3つ数字がでてきて、それぞれ、行数、単語数、バイト数を出力します。圧縮されているファイルの文字数を知りたいだけの時は、`bzcat foo.bz2 | wc -c` などとするのがよさそう。

-L オプションで、ファイルの中で一番長い行の長さが出ます。また、下記の例では、*.c または *.h ファイルのリストから、1行の行数が一番長い行の文字列を表示します。

```
find . -name '\*.[ch]' -print0 | wc -L --files0-from=- | tail -n1
```

特定の文字列だけ何回出現しているか知りたいときはこんな感じでひとつ^{*34}。

```
$ grep -o string file | wc -l
```

6.2 sum

BSD のアルゴリズムで 16bit のファイルのチェックサムと 1024 バイト単位のブロック数を表示するコマンド。`-s` オプションで System V のアルゴリズムを使ってのチェックサムと、512 バイト単位のブロック数を表示。

^{*34} 使いどころがありそうでもない。でもたまに使うことが、あんまりない

```
$ cat /etc/issue
CentOS release 6.6 (Final)
Kernel \r on an \m
$ sum /etc/issue
28978      1
```

6.3 cksum

ファイル名を引数に取ると、CRC のチェックサムを表示します。

```
$ cksum /etc/issue
2950197414 47 /etc/issue
```

6.4 b2sum

coreutils 8.26 から追加されたコマンドです。BLAKE2^{*35} のハッシュ値を出力します。BLAKE2 とは、MD5 や SHA-1、SHA-2、SHA-3 より高速な暗号化ハッシュ関数で、標準 SHA-3 と同程度の安全性があります^{*36 *37}。md5sum のオプションが使えます。b2sum だけに追加されたオプションは -l, --length です。ハッシュ値の長さを変更することができます。ビットで表現するので 8 の倍数である必要があります。--check オプションがある場合は、長さが自動的に決定されるため無視されます。

```
$ b2sum /etc/redhat-release
6e3cdb9cc64ec29b8a11b9f14f6947cb71897404176c874be6d21359035dbc4195af6914d902c553b0b
$ b2sum /etc/redhat-release -l 32
3a140eda /etc/redhat-release
```

^{*35} <https://blake2.net/>

^{*36} って公式サイトに書いてあります

^{*37} <https://tools.ietf.org/html/rfc7693> に RFC があります

6.5. md5sum

6.5 md5sum

128bit のチェックサム(またはフィンガープリントまたはメッセージダイジェスト^{*38})を計算します。リリースするバイナリと、本番でデプロイされているバイナリが一致しているかどうか確かめる時にたまに使います^{*39}。

md5sum が一致するかどうか確かめましょう^{*40}。

```
$ touch a && md5sum a > a.sum  
$ md5sum -c a.sum  
a: OK
```

6.6 sha 系

sha と、sha2 で始まるコマンドをまとめました。原文では、sha1sum と sha2 に分かれています。

sha1sum SHA-1 のダイジェストを計算します。md5sum より安全なダイジェストです。SHA-2 にとってかわられて徐々に廃止すべき、とマニュアルに書かれています

sha2 系コマンド sha224sum, sha256sum, sha384sum, sha512sum というコマンドがあります。それぞれのビット長の SHA ダイジェストを計算します。オプションは、md5sum と同じです。

```
$ sha224sum /etc/issue  
49e10814e2665c2a4040344e927ce4b231152b30c55fb53d8dbb7108 /etc/issue
```

^{*38} この本を読んでいるのにフィンガープリントとメッセージダイジェストを知らないだと!?出直してこい!!と言われないように、知らない人は調べましょう

^{*39} 突然真面目に Tips だしてきたよこの筆者

^{*40} なお、d41d8cd98f00b204e9800998ecf8427e という謎の文字列をググると 47 万件ヒットしました

第 7 章

かぐや様は仕分けたい

ファイルの中身をソート(仕分)するコマンド群です^{*41}。

7.1 sort

ファイル中身をソートするコマンド...と書き始めたかったのですが、それ以外にも機能があります。ファイルを、ソート、マージ、または比較し、表示します。実は3つのモードを持っていて、ソートするモード、マージするモード、ファイルがソートされているかチェックするモードがあります。

チェックオプションのサンプルは下記のようになります。

```
$ seq 12 > k; sort -c k
sort: k:10: disorder: 10
```

マージのオプションはこんな感じです。あらかじめソート済みのファイルを流し込んでやるとソートしてくれます。そのため、seq^{*42}であらかじめ連続したデータを作つておきます。せっかくなのでheadコマンドで表示してみました。

```
$ seq 0 2 10 > a
$ seq 1 2 10 > b
$ head a b
==> a <==
0
```

^{*41} 原題は、Operating on sorted files。元ネタは週刊ヤングジャンプ連載の人気ラブコメ『かぐや様は告らせたい～天才たちの恋愛頭脳戦～』。アニメ4話エンディングは必見。あと藤原書記はめっちゃいい子。間違いない

^{*42} 本書後半で出てきます。連続した数字を出力するコマンドです

7.1. sort

```
2  
4  
6  
8  
10  
  
==> b <==  
1  
3  
5  
7  
9
```

次に、こんなソートを試します。

```
$ sort a b  
0  
1  
10  
2  
3  
4  
5  
6  
7  
8  
9
```

10は後ろに持ってきてたいですよね。そんなときには、-n^{*43} または-g^{*44} または-h^{*45} を付けて下さい。-nの例だとこんなかんじ。

```
$ sort -n a b  
0  
1  
2  
3
```

*43 マイナスがついている数値でもソートしてくれます

*44 マイナスやプラスの記号がついていてもソートしてくれます

*45 human-numeric-sortです。k,G,Mがついていてもソート可能。CentOS5系だとこのオプションは実装されていませんでした。ご注意

```
4
5
6
7
8
9
10
```

そのほか有用と思われるオプションを一気に解説します。

- u 重複を弾いてくれます
- r 逆順に表示してくれます

-k pos1[,pos2] 特定のカラムにある文字列を対象にソートします。 ps aux の 2 カラム目を降順でソートした結果:

```
$ ps aux | sort -nk +2 | tail
```

- t セパレータを設定します。 -k と合わせて使うことが多いです。使用例は複雑なのでマニュアルを見て下さい
- M Month sort です。月の名前でソートしてくれます
- R Random Sort です。 shuf ればいいと思います *⁴⁶
- V Version Sort です。バージョン番号でソートします
- parallel=n n に並行実行数を入れるとパラレルに実行してくれます。ただし、自動で有効なプロセス数が設定されます。あまり気にしないといいでしょう。最大値は 8 です。時代は多コアですからね

7.2 shuf

ファイルを shuffle してくれます *⁴⁷。もし seq をつかって数字をランダムに出したいときはいったん思いとどまって、下記のようにして下さい *⁴⁸。

*⁴⁶ shuf のところで出てくる --random-source が使えます

*⁴⁷ CentOS5.7 な環境でコマンド打ったら出てこない!それもそのはず、CentOS の coreutils のバージョンが古いのでした (5.97)。バージョン 6.4 から新しく加入したコマンドです。

*⁴⁸ 当然、この通りに出てくるわけではありません。--random-source=FILE というオプションもあるのでこだわりたい方はこだわれます

7.2. shuf

```
$ shuf -i 1-4
3
1
2
4
```

同じファイルを共有していれば、同じ結果が返ってきます。つまり、さっき作った k というファイルを使って、あなたと同じランダムな結果を実現してみましょう^{*49 *50 *51}。

```
$ shuf --random-source=k -e hoge fuga piyo choi
fuga
hoge
piyo
choi
```

0-9までの数字を 50 回出すならこれ:

```
$ shuf -r -n 50 -i 0-9
```

コインを 50 回振ってみるならこれ:

```
$ shuf -r -n 100 -e Head Tail
```

k というファイルがなかったらどうすんだって？うーん、2.7 章にこんなコマンドがあります。seed に与える数によって擬似乱数を生成する関数を作つてそれを実行。決して get_seeded_random 関数を単独で実行してはいかん（実際にやつた筆者であった）。

```
get_seeded_random()
{
    seed="$1"
    openssl enc -aes-256-ctr -pass pass:"$seed" -nosalt \
    </dev/zero 2>/dev/null
}
```

^{*49} これを人はランダム、と呼ぶのだろうか。謎である

^{*50} sort にも同じオプションがあります。sort のオプション -R, --random-sort, --sort=random を見てみてください

^{*51} マニュアルには、-r または --repeat というオプションがあります。これを使うと、指定した回数だけ繰り返すので、\$ shuf -r -n 50 -e Head Tail ができます。誰得。なお、Coreutils 8.21 では未実装でした

```
shuf -il-100 --random-source=<(get_seeded_random 42)
```

7.3 uniq

ソート済みのファイルを引数に取ると、重複行を取り除いたデータを書き出してくれます^{*52}。sort にも -u オプションがあり、uniq コマンドを単体で打ったときと同じようなことをやってくれます。よく使うパティーン^{*53}

```
$ cat file | sort | uniq -c | sort -nr | head
```

7.4 comm

2つのファイルを比較して、片方にしかないデータ、両方にしかないデータなどを出力してくれます^{*54}。ベン図を書いて、きちんと整理してデータの集計に当たりましょう。実行結果が独特なのでサンプルを載せます。

```
$ seq 1 3 9 > q
$ seq 1 2 9 > w
$ head q w
==> q <==
1
4
7

==> w <==
1
3
5
7
9
$ comm q w
               1
            3
```

^{*52} テストに出るぞー。そういうえば sort -u とかありましたね？つまり ???

^{*53} file に出現した同じ文字列を出現順にランキング表示です。サーバ管理者でこれが出来なから落第だ！

^{*54} 通話のアブリじゃないですよ。念のため

7.5. ptx

```
4  
5  
7  
9
```

カラムが 3 つあります。単独でカラムを取り出したい場合は、`-1`, `-2`, `-3` というオプションを使います。また 8.26 から `--total` が追加されました。こんな感じです。`total` の行は `-123` とすると単独で取り出せます。

```
$ comm q w --total  
1  
3  
4  
5  
7  
9  
1 3 2 total
```

7.5 ptx

日本語マニュアルによると、「ファイルの内容の整列した索引を生成する」「入力ファイルに含まれる単語の索引を並べ替え、前後を含めて出力します。」⁵⁵ とありますが、使いどころが分からぬッ！

7.6 tsort

前後関係を与えると、その順にソートしてくれます⁵⁶。

実行例を見た方が早いです。`hoge` は `fuga` の前にあるといった組を用意して `tsort` に食わせると順番に並び替えます。ループがあったらどうなるのかな・・・怒られるのかな・・・

```
$ cat text  
hoge fuga  
fuga piyo
```

⁵⁵ http://linuxjm.sourceforge.jp/html/GNU_coreutils/man1/ptx.1.html

⁵⁶ マニュアルによると「有向グラフのトポロジカルなソートを行う」と書かれていて、ちょっと何言ってるかよく分からないです

```
foo bar
bar baz
baz hoge

$ tsort text
foo
bar
baz
hoge
fuga
piyo

$ cat d
a b
b c
c d
d a
$ tsort d
tsort: d: input contains a loop:
tsort: a
tsort: b
tsort: c
tsort: d
a
b
c
d
```

第8章

テーブルの欄操作

8.1 cut

ファイルを垂直に切り出します。オプションが必須のコマンドです。たとえば今月の日曜日の日にちだけ切り出してみましょう^{*57}。`-c1-2` とすると、1から2文字目までが縦方向に切り取られて表示されます。

```
$ cal | cut -c1-2
Su

4
11
18
25
```

csv データから特定のカラムだけ切り出せます。tsort で出てきた text ファイルに対して 2 カラム目だけ表示させてみましょう。文字の区切りはスペース 1 個 (`-d" "`)、2 つめのカラム目を表示 (`-f2`) するオプションを付けます^{*58}。

```
$ cut -f2 -d" " text
fuga
piyo
bar
baz
```

^{*57} `cal` コマンドは今月のカレンダーを表示してくれます

^{*58} [練習問題] `awk` でも同じコマンドを作ってみましょう

```
hoge
```

8.2 paste

ファイルの 1 行 1 行を横にひっつけていきます。具体例はマニュアルに書いてあるので読んで下さい。え？読むのがめんどくさい？しょうがないにゃあ。

```
$ cat num2
1
2
$ cat let3
a
b
c
$ paste num2 let3
1      a
2      b
      c
```

-s (serial) オプションを付けるとこんな感じ。

```
$ paste -s num2 let3
1      2
a      b      c
```

8.3 join

ファイルを横に join します。paste と同じように見えるかもしれません、1 カラム目が共通の 2 つのファイルに対してよしなに join してくれます。

```
$ cat c
00:00 100
00:01 200
00:02 300
$ cat d
00:00 150
00:01 250
```

8.3. join

```
00:02 250
$ join c d
00:00 100 150
00:01 200 250
00:02 300 250
```

とあるサイトのバーチャルホスト別のアクセス数を1分ごとに取るスクリプトを書いて、csvで出してみたりするのがお気に入り。hoge-access.min.logはさっきでてきたファイルcの様な出力になっていて、それを3サイト分、csv形式で出力。あとはexcelにでも突っ込んで1分間ごとのアクセス数を色づけして眺めてみるのが良いのではないでしょか^{*59}。

```
$ for h in `seq -w 0 23` \
> do for m in `seq -w 0 59` \
> do echo $h:$m $(cat hoge-access.log | grep -c $h:$m) ; done ; done | \
> tee -a hoge-access.min.log
$ # などというファイルを三つくらい用意
$ join hoge-access.min.log fuga-access.min.log | \
> join - piyo-access.min.log | \
> sed -e 's/,/,/g' > foo.csv
```

^{*59} ビックデータとかクラウドの時代だとFluentdでなんとかするのが普通かも

第 9 章

キャラクタ操作

9.1 tr

文字の変換と削除を行うコマンド。文字の置換の用途で使うことが多いです。

123 という文字列を、3 を 4 に、2 を 1 に、1 を 6 に変換します。321 という文字列を 456 という文字列に変換するわけではありません。

```
$ echo 123 | tr 321 456  
654
```

ということは、テキストファイルの文字小文字変換もできます。いずれも同じ意味です^{*60}。

```
tr abcdefghijklmnopqrstuvwxyz ABCDEFGHIJKLMNOPQRSTUVWXYZ  
tr a-z A-Z  
tr ':lower:' ':upper:'
```

tr の tips を調べると大体でてくるのは改行の削除なんですが。

```
$ tr -d '\r' < dosfile.txt > unixfile.txt
```

9.2 expand

タブを 8 つのスペースに変換します。おしまい^{*61}。

^{*60} ファイル名を小文字にしたい？mv のコマンドを作って実行すればいいのだ★

^{*61} [練習問題] 同じことを sed あるいは他のコマンドで代用してみよう

9.3 unexpand

スペースをタブに変換します。スペースが乱雑に現れてもなんとかしてくれそうです^{*62}。

^{*62} [練習問題]同じことを sed あるいは他のコマンドで代用してみよう

第 10 章

ファイルリスト表示

10.1 ls

ディレクトリの中身を表示します。奥深いコマンドです^{*63}。さて問題です。`ls` を単独で打ったときはどのような挙動をするでしょうか。おそらくこのような本を買っているということは、説明する必要がないのかなと思いつつ^{*64}。個人的によく打つオプションは、`ls -lhatr` です。オプションをさらっとおさらいしましょう。

どんなファイルを表示するか

- a . から始まるファイルも表示します
- A . から始まるファイルを表示しつつ、. . .^{*65} . . .^{*66} は表示しません
- B ~で終わるバックアップファイルを表示しません
- d ディレクトリのみを表示します^{*67}
- I pattern \$ ls -I “*i*” とすると、i を含むファイルやディレクトリが表示されなくなります
- R ディレクトリを再帰的に表示。下手すると大変なことになるので注意

^{*63} マニュアルでは 7 つのセクションに分かれています

^{*64} 解説。カレントディレクトリの中身を表示します。ただし、ディレクトリの中身を再帰的に表示しません。また、. から始まるファイルも表示しません。アルファベット順で表示します。リストの結果が端に到達したら縦方向に並べます。画面に表示出来ない文字は ? で表示します

^{*65} カレントディレクトリ

^{*66} 一つ上のディレクトリ

^{*67} ls してたくさん普通のファイルがある中でディレクトリだけを表示したいときに使う

10.1. ls

どんな情報を表示するか

--full-time	フルなタイムを表示します。stat でいいような...
-g	ファイルの owner が省略され、group のみを表示します
-G	ファイルの owner を表示し、group は表示しません。GNU でないバージョンの ls の互換性のためのオプション
-i	inode 番号を表示します *68
-h	無味乾燥な数字の羅列であるファイルのサイズを読みやすくしてくれます。ひゅーまんりりーだぶるの h です
-l	ファイルのパーミッションやハードリンクの数、owner group、ファイルサイズ、タイムスタンプを表示します

ちなみに： ハードリンクの数

-l を付けたときこんな感じで表示されます。k というファイルを作っておきます。

```
$ ls -l k  
2875312 -rw-r--r-- 1 nanaka nanaka 27 Nov 29 03:19 k
```

このとき、nanaka の前の 1 ってのはなによ、という問題。こうすると分かります

```
$ ln k l # ハードリンクを張ります。同じ inode を指すファイルを作ります  
$ ls -l -i k l  
2875312 -rw-r--r-- 2 nanaka nanaka 27 Nov 29 03:19 k  
2875312 -rw-r--r-- 2 nanaka nanaka 27 Nov 29 03:19 l  
  
$ rm k # k ファイルを消すと...?  
$ ls -l -i l  
2875312 -rw-r--r-- 1 nanaka nanaka 27 Nov 29 03:19 l # l になった!
```

-n	ファイルのグループ、オーナーを数字で表示します *69
-o	-G とおなじ
-s	ファイルに対するディスクの割当量を表示します *70

*68 inode ってなに？ググりましょう

*69 see /etc/passwd

*70 手元の環境だと、小さなファイルに関しては 4 kbytes が割り当てられていました

ソート順を指定

-c	ファイルが作られた (ctime) 順でファイルを表示します
-f	ソートせずそのまま表示します。-a が有効、-l, -color,-s が無効になっています
-r	逆順にソート
-S	ファイルの大きさ順にソートします。デフォルトは大きい順に並びます。小さい順に並べるなら、-rs
-t	ファイルの更新時間 (mtime) 順にソートします
-u	ファイルにアクセスした時間 (atime) 順にソートします
-U	ファイルのソートを行いません。ファイルがたくさん入っているディレクトリで効果を発揮するでしょう
-v	バージョンや番号順に表示します。1.2.3 と 1.2.10 を意図したとおりに並べたいときにオススメ *71
-X	拡張子のアルファベット順で表示します。こんなオプション知らなかつたぜ

表示のフォーマットを指定

-1	1 ファイル 1 行で表示します。ファイルの一覧をファイルに書き出すときに使います
-C	ファイルを縦方向に表示します。デフォルトの動作です
--color	表示の際の色を決めます。-color=auto が alias にあるのが普通 *72。環境変数の LS_COLORS に色が定義されていますが、いつみても呪文だなあと思います
-F	ファイル名の一番最後にファイルタイプを示す 1 文字をひっつけます。/はディレクトリ、@はシンボリックリンク、>はソケットファイルなどなど
--file-type	-F ぽいけど実行可能ファイルに関してはファイルタイプを示す 1 文字がつきません *73
--indicator-style=word	word に、none,slash,file-type,classify のいずれかを入れると、それぞれ、デフォルトの動作、-p オプション、-file-type

*71 Coreutils のセクション 10.1.4 にどのようにソートするのか書かれています

*72 ディストリビューションによる？なぜ疑問系？

*73 [問題] 実行可能ファイルのファイルタイプを表す 1 文字はなんでしょう

10.1. ls

	オプション、-F オプションと同じ意味になります
-k	1024 バイト単位でブロックサイズを表示します。 -h とか付けると効果がなくなります
-m	ファイルをだらだらっとカンマ区切りで表示します
-p	ディレクトリの後ろに/を付けます。 そういえば、あなたのデフォルトの ls の動作はどうでしたっけ？ * ⁷⁴
-x	たくさんのファイルが入っているディレクトリを表示したとき、横方向にファイルをソートします。
-T cols	横に並べるファイルの数を指定。 -T 1 にするとファイルの一覧が改行されずに 1 行で表示しようとするので画面が崩れること請け合い
-w cols	横方向にどれだけ表示するか。 -w 1 とかすると-1 と同じ効果 * ⁷⁵

タイムスタンプの表示形式

--time-style=style	タイムスタンプのフォーマットを指定できるよ！やったね！
	* ⁷⁶

ファイル名の表示形式

-b	例を見てみましょう
-----------	-----------

```
$ touch Ctrl-v Enter # ctrl を押しながら
$ # v を押しキーボードから手を離す
$ # 一呼吸おいて Enter を押す。さらにもう一度 Enter
$ ls
?
$ ls -b
\�
```

ファイルの消し方は自分で考えてね！

*⁷⁴ 読者をゆさぶる筆者の図。多分口元が緩んでいるかもしれないし、そういえば自分の ls の動作ってどうだったっけ？と自分ではまっている

*⁷⁵ マニュアルには引数ないことになってるけど、実際は必要

*⁷⁶ ふええ、使いどこが分かりません！

- N** ファイル名をクオートしません。危険が危ない
- q** 改行とか表示できない文字を?で表示。デフォルトの動作です
- Q** ファイル名を”“で囲みます
- show-control-chars** 表示できない文字もそのまま表示します。デフォルトの動作です

余談ですが、ls を削除してしまった環境でファイルの一覧を得るなら echo *。

10.2 dir

ls -C -bと同じ。

10.3 vdir

ls -l -bと同じ。

10.4 dircolors

ls のカラー設定^{*77}。呪文なので唱えて下さい。実行方法が特殊

```
$ eval "$(dircolors [option]... [file])"
```

-p オプションで設定を見ることが出来ます。

^{*77} ぶっちゃけた話、実機のコンソールに入ることは滅多にないので ssh クライアントで色を設定すればよくね?とは思っている。え? Mac?自分で何とかして下さい...

第 11 章

基本的操作

11.1 cp

ファイルやディレクトリをコピーします。コピーするだけなら問題ないんですが… というところ。細かい仕様を把握しておかないと事故が起きるので^{*78}、もし本番環境でやるなら事前テストすることをおすすめします。とくに、* や、末尾 / のあるなしで効果が変わります。さて、もう少し細かい挙動を見て行きましょう。原文では、デフォルトだとディレクトリはコピーしないけど、-R,-a,-r オプションをつけると、ディレクトリもコピーするようになるよ、だそうです。シンボリックリンクからコピーするときは、そのシンボリックリンクしかコピーしないけど、-archive(-a),-d,-dereference(-L),-no-dereference(-P),-H オプションがあるとデフォルトの設定を上書きしちゃうよ、とのこと。

オプションを解説します。

-a,--archive ファイルの属性と構造をそのままコピーします。というのも、デフォルトでは、コピーしたファイルは、コピーした時点のタイムスタンプになります。

-b,--backup[=method] バックアップオプションです。上書きや削除が発生する場合に使います。-b の場合は引数を取りません。

-d,--no-dereference シンボリックリンクをコピーするとき、シンボリックをコピーします。ハードリンクの場合も同じくハードリンクとしてコピーします

-f,--force コピー先に同じファイルがあっても上書き…と思いまして実際の解説はこうなっています^{*79}。コピー先の削除またはアンリンクするために、ファイルを開こうとします。それができない場合、削除とオープンを再度試します

^{*78} すでに存在するディレクトリの中に、コピーするディレクトリのファイルをぶちまけて混ざる事案や、シンボリックリンクを考慮するかしないか事案など

^{*79} 日本語 manpage と coreutils を適当に混ぜあわせて解説を書いてます

-H コマンドで指定されたすべてのシンボリックリンクを辿ります

-i,-interactive インタラクティブです。ファイルを上書きするとき、上書きするかどうか聞きます。

環境によっては、alias cp -i されています。逆にうざかったりします

-l,-link ハードリンクをつくります。ディレクトリは引数にとれません

-L,-dereference シンボリックリンクをすべて辿り、それが指しているファイルやコピーを上書きします

-n,-no-clobber 上書きしないオプションです。-n の前に-i オプションがあった場合は-i オプションを無効化します

-P,-no-dereference 日本語の manpage^{*80} と説明が食い違っているのは内緒にしておきましょう。原文では、シンボリックはシンボリックとしてコピーするとあります

-p,-preserve[=attribute_list] ファイルの属性を保ちつつコピーを行います。-p オプションで、すべての属性を保ちます。ファイルのオーナー、タイムスタンプなどコピー時に変更しない属性(attribute_list)を選択できます。例えば、cp --preserve=ownership,timestamp origin copyfile といった感じです。逆にこの属性は保存しなくてもいいや！というときに、--no-preserve[=attribute_list] というオプションもあります。

-R,-r,-recrusve ディレクトリを再帰的にコピーします。-R であっても-r であっても効果は同じです

--reflink copy-on-write(COW) を使えるシステムであれば COW を使います。詳細は wikipedia で

-s,-symbolic-link ファイルに対してシンボリックリンクを作成します。ディレクトリのシンボリックリンクは作ることができません。ln -s の威儀は保たれました。あ、あとシンボリックリンクをサポートしていないシステムでエラーが出るかも(・ω<)

-u,-update 修正時刻がコピー元と同じかそれより新しい場合、コピーをしません

-v,-verbose verbose です

^{*80} <http://linuxjm.sourceforge.jp/html/gnumaniak/man1/cp.1.html> 日本語ドキュメントでは、-P と --parent が同じであると書かれています。試してみたところ -P オプションについては、原文が正しい模様。--parent オプションの説明としては合っていました

cp の速度

cp の速度と進捗状況が知りたい時があります。こんなときは、pv(pipe viewer) を使いましょう。yum の extra 経由でインストールするか、公式サイトから RPM を落としてきてインストールします。pv の実行例です：

```
$ pv coreutils-8.22.tar.xz > hoge
5.09MiB 0:00:00 [ 657MiB/s] [=====>] 100%
```

11.2 dd

ファイルのコピーとか変換とか行います^{*82}。ファイルと言っていますが、デバイスにも対応しています。よくあるディスクのコピーはこんな感じ

```
# dd if=/dev/sda1 /dev/sdb1
```

「変換」はどこいったんじゃ！というツッコミもあるかと思いまして、例を挙げます。text というファイルを入力に使います。

```
$ dd conv=ucase if=text of=test2
```

これで text ファイルの中身のアルファベットが大文字になります！やったね！！^{*81}
あと、これまで dd がいつ終わるかわからない！という声に応えて status=progress というオプションが 8.24 から入りました。コマンド終了時の出力で、でかい数字のバイト表記で単位がよくわからん！という声に応えて、適切な SI 単位が 8.25 から入るようになりました。
3441325000 bytes (3.4 GB, 3.2 GiB) copied こんな感じです。

11.3 install

ファイルの属性を指定しつつファイルをコピーすることが出来ます。ファイルをコピーしてからオーナーを変更するという動作がこのコマンドだけでできます。例をいくつか。まずは、file というファイルを、ユーザ名 user、グループ名 group で/tmp ディレクトリにコピーする例：

^{*82} なぜ dd という名前なのは、'Dataset Definition' の略だとか、'Convert and copy a file' の略で cc にしたかったけどすでにそのコマンドがあったので dd にしたとか。真相は自分で確かめよう！

^{*81} ucase 以外にも、ebcdic や ibm といったオプションもあります

```
# install -o user -g group file /tmp/
```

ディレクトリも作ってくれます。mkdir -p 早いこともやってくれます:

```
# install -o root -g root -m 755 -d /hoge/fuga/piyo/choi
```

11.4 mv

ファイルの移動を行います。副次的な作用としてファイルの名前を変えることが出来ます。cp のオプションと似ていますので、細かい説明は省略します。

11.5 rm

「あのころのきおくをけすにはどうやるです?」という妖精さんからの質問に答えましょう。過去にだーくふれいむますたーであった頃の黒歴史を消去するコマンドです^{*83}。

前回 ls コマンドの説明の時に作った、改行コードがファイル名になっているファイルを消してみましょう。

```
$ ls -b
\rm
$ rm Ctrl-v Enter # ctrl を押しながら v を押しキーボードから手を離す
$ # 一呼吸おいて Enter を押す。さらにもう一度 Enter
```

消したファイルの一覧を出す例。

```
$ shuf -i 1-10 | xargs touch
$ ls
1 10 2 3 4 5 6 7 8 9
$ rm -v *
removed `1'
removed `10'
removed `2'
removed `3'
removed `4'
removed `5'
```

^{*83} 黒歴史をバージョン管理していたら別

11.5. rm

```
removed `6'  
removed `7'  
removed `8'  
removed `9'
```

rm -rf /

一般ユーザでスクリプトを実行したとき、空の変数\$hoge を対象に rm -rf \$hoge ってやってユーザディレクトリの中身を消すってことは、たまにあります。手動で実行する機会はめったにないとは思いたいところではありますが、不運にして root で rm -rf / を実行してしまった場合、どうなるかについて。突然シャットダウンするといったことはありません。/proc ディレクトリが残るようです⁸⁴⁸⁵。さて、突然の rm -rf / を防ぐにはどうするかというと、rm の alias に --preserv-root をつけておきましょう。ついでに、safe-rm というソフトウェアがあります。これは、rm のラッパであり、システム的に消してはいけないディレクトリを消そうとするとメッセージが現れ消さずに済みます。

⁸⁴ 筆者も実際に VMware 上で rm -rf / をやったことがあります。コマンドが実行できないだけで OS 自体は起動している状態でした

⁸⁵ 参考 <http://katsu.watanabe.name/doc/rmrf/>

rm undo

いやー、気持ちは分かるんですけどね。わかりますよ。非常によくわかります。undo みたいなコマンドを打てば元通りにして欲しいですよね？大丈夫です。自分でコマンド作ってください。というのは半分冗談で、現実的な答えとしては、ゴミ箱を作ることです。ゴミ箱ディレクトリを作って、mv にエイリアスを貼ります。alias rm='mv --target-directory=\$HOME/.Trash' などとします。結論をいうと、バックアップ取れ！これに尽きます。

ext3grep

ext3grep というツールがあります。ext3 ファイルシステムから文字列を grep してくれます。公式ドキュメントいわく、「unmount the partition ASAP ; do not mount it again.」だそうです。ん？バイナリファイル復活させたい？がんばれ！

rm には alias が貼ってあることがありまして、alias rm と打つと、alias rm='rm -i'

と出てくることがあります。`-i` はこのファイル本当に消す？と聞いてくるオプションです。このエイリアスの呪縛から逃れるには、`\rm` とバックスラッシュを付けます。さっきの alias のとき、`rm -rf file` とやると、`rm -i -rf file` となります。オプションの順序の関係で、あとに置かれたオプションで上書きされるので、`-i` オプションは上書きされます。ということで、本当に消す？って聞いてこなくなります。

11.6 shred

爆ぜたり弾けたりしていた黒歴史を強力に清算するコマンドです *86。ファイルやファイルシステム (`/dev/sda3` など) を引数にとり、何度も上書きを行って黒歴史を清算し、復元される可能性を減らします *87。

原文を読んでいきましょう。`rm` しただけでは、実際にはファイルは消えてはいません。ファイルの登録情報が消えるだけです。さらにデータが上書きされても、痕跡を読み出す機械でデータを復元されてしまうことがあります。復元できないようにするには、酸で溶かすことです。フロッピーディスクのようなメディアならば良い方法ですが、ハードディスクでは難しいため、`shred` は酸で溶かさずとも同様の効果を発揮しようとします。

ただし、ファイルシステムによっては、その方法が通用しないことがあります。NFS 経由などでスナップショットをとっていたり、圧縮されているファイルシステムなどです。

使い方はこんな感じです。ファイルを削除したい場合は、`-u` オプションを付けてください *88。

```
$ shred -u secretFile
```

デフォルトでは 3 回上書き処理を行います。変更したい場合は `-n <回数>` を指定してください。1 回だけ上書きするのであれば下記のようにしてください。`-v` は進捗を表示します。

```
# shred -v -n1 /dev/sda5
```

SSD に対して下記のように、0 のデータ (-z) を書き込むとき、ディスクのコントローラで最適化されてしまい書き込みがブロックされてしまうことがあるので注意 *89。

```
# shred -v -n0 -z /dev/sda3
```

*86 劇場版でも爆ぜたり弾けたり。末永く爆発して欲しいですね（おっさん脳

*87 ただし、人々の記憶には残っているでしょう。物理破壊がより有効です（ハードディスクのことです

*88 じつは-u オプションに種類があって、デフォルトだと `wipesync` です。気になった人はマニュアルへ Go!

*89 原文で SSD が出てくるとは思わなかった

第 12 章

スペシャルファイルタイプ

シンボリックリンクや FIFO、ディレクトリなどのファイルの操作を行います。

12.1 link

link システムコールを経由してハードリンクをつくるコマンドです。link コマンドより次の ln コマンドの方が一般的によく使われます。実行例を示します。

```
$ link filename linkname
```

12.2 ln

ファイル間にリンクをつくるコマンドです。ファイルやディレクトリに対して、ハードリンク(デフォルトの動作)や、シンボリックリンク(-s オプション)を作ることができます。オプションの順番は 実体のあるファイル その後に リンクするファイル名 を指定します。

```
$ ln -s ファイル シンボリックリンクのファイル
```

主なオプションです。

- s 頻出！ シンボリックリンクを作ります。デフォルトだとハードリンクを作ります。違いについては自分で調べよう！
- d, -F, --directory ディレクトリのリンクを作ります。システムによっては失敗します
- f リンクするファイル名を削除します。ディレクトリの場合、上書きできず、シンボリックリンクディレクトリの下にシンボ

	リックリンクが出来ているというオチが待っています
-i, --interactive	インタラクティブになる
-L, --logical	シンボリックリンクからハードリンクを作成します。何言つ てるかわかんないけどそうなんです。やってみて
-P, --physical	シンボリックリンク自体へのハードリンクを作成します。な んのこっちゃわからなくなってきたね
-r, --relative	相対パスでリンクを作成します。ある意味で便利ですが多用 すると大変なことに

12.3 mkdir

ディレクトリを作ります。押さえておくべきオプションは二つ。パーミッションを指定する `-m`、存在しない 2 階層以上のディレクトリを作る `-p` です^{*90}。なお `-p` オプションを付けると、存在しているディレクトリを再度作ってもエラーになりません。つまりディレクトリがあるかどうかの判定を省略できます。地味に便利。

```
$ mkdir -m 777 dir
$ ls -ld dir
drwxrwxrwx 2 nanaka nanaka 4096 7月 27 01:28 dir
```

12.4 mkfifo

名前付きパイプを作ります。パイプとは、`|` です^{*91}。具体例を示します。

```
$ mkfifo pipe
$ ls -l > pipe &; cat < pipe
```

2 行目のコマンドは、わざとワンライナーで書いていますが、別のセッションで試すと感動が増します。`;` の前後のコマンドを逆にしても動作します^{*92}。

^{*90} `mkdir` とは関係ないけど、自分がつくったディレクトリに `chmod -x directory`とかしちゃ駄目だぞ！おっちゃんとの約束だ！(やったことない人はやってみようというフラグ)

^{*91} `|` は、名前なしパイプ (unnamed pipe) です

^{*92} “Introduction to Named Pipes” <http://www.linuxjournal.com/article/2156>

12.5 mknod

FIFO や、キャラクタースペシャルファイル、ブロックスペシャルファイルを作ります^{*93}。

```
$ mknod --help
Usage: mknod [OPTION]... NAME TYPE [MAJOR MINOR]
```

キャラクタースペシャルファイルとは、キーボードやマウスなどの入力や出力を扱うファイルです。キャラクタースペシャルファイルは 1 バイトずつの読み出しだけですが、ブロックスペシャルファイルはある程度の塊としてデータを取り扱います。下記、一番はじめの b がブロックスペシャルファイル、c がキャラクタースペシャルファイルです。それぞれハードディスク、zero です。

```
$ ls -l /dev/hda
brw-rw---- 1 root disk 3, 0 May 22 2012 /dev/hda
$ ls -l /dev/zero
crw-rw-rw- 1 root root 1, 5 May 22 2012 /dev/zero
```

MAKEDEV というコマンドでもデバイスファイルを作ることが出来ます。

12.6 readlink

シンボリックリンクを引数に与えると、絶対パスを表示します。例えば、PID から実行されているバイナリの絶対パスを得る方法はこちら^{*94}。

```
$ readlink -f /proc/$pid/exe
```

12.7 rmdir

空のディレクトリを削除します。普段は rm -rf を使うところ。なお、--ignore-fail-on-non-empty オプションをつけることによって、中身の入っているディレクトリでもエラーを返さず、削除もしません^{*95}。

^{*93} NAME,TYPE,MAJOR,MINOR,MINOR に当たる部分は、カーネルソースのドキュメントか <https://www.kernel.org/doc/Documentation/devices.txt> を参照

^{*94} <http://www.commandlinefu.com/commands/view/11820/bin-file-of-a-pid>

^{*95} 本書の執筆をサポートしてくれた mtgto 氏によると、ホームディレクトリで、 rmdir --ignore-fail-on-non-empty .ssh を実行したところ、警告もなくコマンドが終わってしまった ssh ディレクトリが消えてしまったと錯覚する事案が発生したこと。みなさんも気をつけましょう

12.8 unlink

システムが提供している unlink を使ってファイルを削除します。

第 13 章

ファイルの属性を変更

13.1 chown

ファイルのオーナーとグループを変更します。`--reference=filename` で `filename` とそっくりのオーナーとグループになります。シンボリックリンクを追うかどうかのオプションもあります。ファイルのオーナーを変えるので基本的に root(あるいは sudo) で操作。ちなみに、`owner` と `group` のセパレータは `:` が一般ですが、筆者は `.` 派。

```
# chown root:root rootfile # セパレータは、: でも . でも
# chown user. userfile    # 動作します
```

13.2 chgrp

ファイルのグループを変更。こちらにも `--reference` オプションがあります。

ちなみに: uid や gid で指定する方法

ご存知のように、`chown` や `chgrp` は、変更するユーザ名やグループ名を引数に与えると指定したユーザ名やグループ名を変更することができます。uid とか gid で指定できたら、うれしくない?え?できちゃうの?出来ちゃうんですねこれが。uid や gid の数字の前に + を入れれば良いのです。

```
# chown +1000.+1000 hoge-file
# chgrp +$numeric_group_id fuga-file
# chown +0:+0 /tmp/root-file
```

Coreutils のマニュアル 2.6 章に書いてあります。man 引いても出てこないです。ユーザ名が数字だったときの対処のため、uid,gid を指定するときは “+” を付けます。ちなみに + はユーザ名やグループ名に使えません。実際に実行してみると「useradd: invalid user name ‘love+’」だそうです。Solaris 10 は例外。

13.3 chmod

ファイルのパーミッションを変更します。数字でパーミッションを指定できたり、相対的な感じで other のみ read を取り除くこともできます。

```
# touch hoge
# chmod 777 hoge
# ls -l hoge
-rwxrwxrwx 1 root root 0 7月 27 01:56 2014 hoge

# ls -l file
-rw-r--r-- 1 root root 0 7月 27 01:53 2014 file
# chmod o-r file
# ls -l file
-rw-r----- 1 root root 0 7月 27 01:53 2014 file
```

MODE の指定はこんな感じです。あとは流れで *⁹⁶

Each MODE is of the form ` [ugoa]*([-+=] ([rwxXst]*|[ugو]))+'.

13.4 touch

呼吸を止めて一秒なコマンドです *⁹⁷。ファイルのタイムスタンプを変更するコマンドです。中身の無いファイルを作ることも出来ます。ファイルの atime,mtime を任意に変更するオプションもあります。時間の指定の方法は必要であれば調べましょう *⁹⁸。ここでも --reference オプションが使えます。

*⁹⁶ おい

*⁹⁷ 違います

*⁹⁸ date コマンドの日付フォーマットとも違っていて若干もによる (-t オプション)。 --date= オプションで date コマンドの --date オプションと同じ指定ができます

第 14 章

このディスク容量には問題がある！

ディスク使用状況 (Disk usage) です。原文曰く、ディスクは無限のデータ容量を保持できない、だそうです。確かに無限の容量があったら必要ありませんね。`du` くらいは残して欲しいところ。将来、`df` コマンドをたたく必要がなくなる日は来るのか *⁹⁹。

14.1 df

ディスクの空き容量を示します。よく使うオプションは、`df -h` です。ディスクの使用量、空き容量が GB や TB 単位で出ます。たまに使うオプションは、`df -i` です。inode の使用量を表示します。ファイルをフォーマットするときに inode 数が足りるかどうか、心にとめておくといいことがあるかもしれません。そして、inode 枯渇はしばしば深刻な問題を引き起します。ファイルシステムの形式 (ext3 や tmpfs など) を表示するときは、`df -T` とします。

実行例です。`-o(--output)` オプションでいろいろ見れます。必要なカラムだけ表示することもできます。また、例には示しませんが、`-l` でリモートなマウントは表示しないオプションです *¹⁰⁰。

# df	Filesystem	1K-blocks	Used	Available	Use%	Mounted on		
	devtmpfs	241308	140	241168	1%	/dev		
	tmpfs	251108	0	251108	0%	/dev/shm		
	/dev/vda1	20511356	2025232	17437548	11%	/		
# df --o	Filesystem	Type	Inodes	IUsed	IFree	IUse%	1K-blocks	Used
	devtmpfs	devtmpfs	60327	529	59798	1%	241308	140
	tmpfs	tmpfs	62777	1	62776	1%	251108	0

*⁹⁹ 元ネタは、この美術部には問題がある！

*¹⁰⁰ 8.21あたりでアクセス出来ないリモートマウントがあるときはハングしたそうですが、8.24で修正

```
/dev/vdal      ext4      1310720 65729 1244991      6%  20511356 2025232
# (続き)
  Avail Use% File Mounted on
241168   1% - /dev
251108   0% - /dev/shm
17437548 11% - /
```

14.2 du

カレントディレクトリにあるファイルのサイズをすべて表示します。 `du -h` さえ覚えていればなんとかなります。 `-h` は、ひゅーまんりりーだぶるの `h` です。個々のファイルサイズはいらないよ、というときは `summarize` オプションをつけて `du -hs` で所望の結果。`du -h /home/*` こういうことをすると、誰が一番ディスクを使っているかランキングをとることです^{*101 *102}。筆者が一番使うオプションは、`du -hcs *` です。`*`をつけると、カレントディレクトリにある各ファイルとディレクトリの容量を表示してくれます。`-c` は、トータルの容量を教えてくれます。

14.3 stat

ファイルが作られた日時や編集された時間を表示するコマンド、と思いきや、ファイルのあらゆる属性を表示するコマンドです。と、思いきやほとんど `ls` で事足りるのでした。`stat` でとれる属性は、割り当てられているブロックサイズや `inode` 番号、`atime` のエポックタイム数値などが取得できます。API的に取得するにはちょうどいいコマンドです^{*103}。

```
$ stat hoge
  File: 'hoge'
  Size: 5335124          Blocks: 10424          IO Block: 4096   regular file
Device: ca01h/51713d     Inode: 114           Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 500/ec2-user)  Gid: ( 500/ec2-user)
Access: 2014-04-17 19:16:38.329808922 +0000
Modify: 2014-04-17 19:21:24.705872432 +0000
Change: 2014-04-17 19:21:24.705872432 +0000
 Birth: -
```

^{*101} さあここで `sort` の出番ですね

^{*102} ファイルがでかいと、i/o を食うので注意

^{*103} Birth ってなんでしょうね？

14.4 sync

メモリにバッファされているデータをディスクに書き込みます。サーバを halt、 reboot あるいは shutdown する前に sync; sync; sync するという文化で年齢が分かるかもしれません。なお、 --help, --version 以外のオプションは無視されます^{*104}。

14.5 truncate

ファイルのサイズを減らしたり増やしたり。ボクと契約して 10M のダミーファイルを作ってよ！と言われても慌てず騒がず truncate -s 10M file して提出して下さい^{*105}。ファイルの中身を空にすることもできますが、 \$ > file でいいよねという風潮 [脳内調べ]。

^{*104} と、思いきや 8.24 からオプションが追加されました。マジで！？こんなマイナーなコマンドに！？ -data と -file-system です。ファイルを指定して、どのように sync(sync, syncfs, fsync, fdatasync) するのか指定できます。

^{*105} [問題] 提出したデータの中身はどうなっているでしょうか。確認してみましょう

第 15 章

文字を表示

文字を表示するコマンドです。

15.1 echo

与えられた文字を標準出力に書き出します。デフォルトだと、最後に改行が入るので、ハッシュ値を作るときには注意して下さい。改行が入らないようにするために、`-n` オプションを。`\n`(new line) といった特殊文字を出力するためには下記のように。

```
$ echo -e "a\nb\nc"  
a  
b  
c
```

15.2 printf

C 言語の `printf` に似たフォーマットで文字列を出力します。たとえばこんな感じ

```
$ printf "%d" "'a"  
97
```

15.3 yes

`Ctrl-c`(`ctrl` を押しながら `c` を押す。つまり `kill` が実行) されるまで、引数に渡された文字列を延々と表示します。最後は `kill` される宿命なので、終了コードは必ず 1 になります。「イエス、

15.3. yes

アスミス」はこのようにしてください。

```
$ yes asumisu
```

host というホストへのネットワークの速度を測りたい場合はこちらです^{*107 *108}。cp のところででてきた pv^{*106} が再登場します。

```
yes | pv | ssh host "cat > /dev/null"
```

^{*107} <http://www.commandlinefu.com/commands/view/4434/live-ssh-network-throughput-test>

^{*108} yes というかマシンの性能によるんじゃないかなあ、などと思う今日この頃

^{*106} pipe viewer

第 16 章

条件

原文は Conditions。

16.1 false

何もしない、成功しない。戻り値は 1 です^{*109}。\$? は、直前に実行したコマンドの戻り値を拾ってくれる変数です。

```
$ false ; echo $?
1
```

16.2 true

何もしない、成功。戻り値は 0 です^{*110}。シェルスクリプトの if 文で、何もしないときに使います。true と同じ意味のビルトインコマンド : で代用することができます。

```
$ true ; echo $?
0
```

ここで終わるのも何なのでソースを見てみましょうか^{*111}。main の部分です。

^{*109} C 言語と違うので混乱します

^{*110} 混乱してきましたね。実際 if を実行するときは、脳内で真か偽かで判断しましょう。1 か 0 とか考えていると混乱します

^{*111} <https://github.com/coreutils/coreutils/blob/master/src/true.c>

16.2. true

```
int
main (int argc, char **argv)
{
    /* Recognize --help or --version only if it's the only command-line
     * argument. */
    if (argc == 2)
    {
        initialize_main (&argc, &argv);
        set_program_name (argv[0]);
        setlocale (LC_ALL, "");
        bindtextdomain (PACKAGE, LOCALEDIR);
        textdomain (PACKAGE);

        /* Note true(1) will return EXIT_FAILURE in the
         * edge case where writes fail with GNU specific options. */
        atexit (close_stdout);

        if (STREQ (argv[1], "--help"))
            usage (EXIT_STATUS);

        if (STREQ (argv[1], "--version"))
            version_etc (stdout, PROGRAM_NAME, PACKAGE_NAME, Version, AUTHORS,
                         (char *) NULL);
    }

    return EXIT_STATUS;
}
```

EXIT_STATUS は、ご覧のとおり。

```
#ifndef EXIT_STATUS
#define EXIT_STATUS EXIT_SUCCESS
#endif
```

おっと、ここで、false のソースコード見てみましょうか *¹¹²。

```
#define EXIT_STATUS EXIT_FAILURE
#include "true.c"
```

*¹¹² <https://github.com/coreutils/coreutils/blob/master/src/false.c>

16.3 test

コマンドの戻り値を判定して条件分岐します。コマンドとしては、`$ test expression` や、ビルトインコマンドとして [expression] が利用できます。expression については、shell のマニュアルに書いてあります。指定されたファイルが存在するか、数値の大小比較などができます。

```
$ HOGE=str
$ if [ "xstr" = x$HOGE ] ; then
>   echo $HOGE
> else
>   echo $HOGE is not str
> fi
str
```

`HOGE` という変数が `str` かどうかを比較するサンプルです。もし、`x` がなくて、`$HOGE` が空だと [str =] となってしまい、syntax error になるので慣習として `x` を付けています。[] の返値が 1 か 0 で条件分岐します。つまり、["xstr" = x\$HOGE] というコマンドが実行可能ですが *¹¹³。そんなわけで、[というコマンドがあるんですよ...もちろんコマンドなので、[のあとにスペース入れないといけませんよ... きこえますか... きこえますか... あっ、見られてますね... *¹¹⁴ expression の比較のサンプルです。なぜ `eq` とか `ne` とかしてしまったんや。> にするとリダイレクトに食われるからか。仕方ないね。

```
$ ONE=1
$ TWO=2
$ ICHI=1
$ test $ONE -eq $ICHI ; echo $? # 数字が一致しているとき真 (equal)
0 # 真
$ test $ONE -ne $ICHI ; echo $? # 数値が違うとき真 (not equal)
1 # 偽
$ test $ONE -gt $TWO ; echo $? # ONE > TWO のとき真 (grater than)
1 # 偽
$ test $ONE -lt $TWO ; echo $? # ONE < TWO のとき真 (less than)
0 # 真
```

`file,file1,file2` というファイルがあった場合は、

*¹¹³ 補足しておくと、`test "xstr" = x$HOGE` というコマンドと等価です

*¹¹⁴ とくにオチはない

16.4 [

```
$ touch file1
$ touch file2
$ test file1 -nt file2 ; echo $? # file1 が file2 より新しいとき真 (newer than)
1 # 偽
$ test file1 -ot file2 ; echo $? # file1 が file2 より古いとき真 (older than)
0 # 真
$ test -e file ; echo $? # file が存在するとき真
0 # 真
```

文字列のとき

```
$ STR=string
$ test $STR ; echo $? # $STR になにか入っていたら真
0 # 真
$ test $STR = $STR ; echo $? # 文字列比較。==でも可
0 # 真
$ test $STR != $STR ; echo $? # 文字列不一致
1 # 真
```

なお、expression の先頭に ! をつけると否定、 expression -a expression の -a は AND 条件、 同様に -o は OR 条件になります。ファイルのタイプ(スペシャルファイルか、シンボリックリンクか、ディレクトリか、ファイルかどうかなど)を判定することもできます。

16.4 [

man はありませんが、コマンドとして存在します。ソースもあります。貼り付けとりますね *¹¹⁶

```
#define LBRACKET 1
#include "test.c"
```

16.5 expr

式を評価します。といっても最近はもっぱら \$() や \${()} を使っています。例は、join の時に出てきています。括弧二つの方は何となく数値計算ができるので電卓代わりに使っています。echo \$((12*34)) といった感じです *¹¹⁷。

*¹¹⁶ <https://github.com/coreutils/coreutils/blob/master/src/lbracket.c>

*¹¹⁷ そして始まる bc との宗教戦争

第 17 章

リダイレクション

シェルのリダイレクションです^{*118}。

17.1 tee

出力を複数のファイルやプロセスに渡すコマンド。`tee` は T のことで、T 型に出力という意味です^{*119}。コマンドの結果をファイルに書き込むときよく使うリダイレクション `> file` のとき、何が出力されるのか表示されません。ファイルにも書きつつ、標準出力にも出力するときに使います。`join` コマンドのところでも本コマンドが登場するのでご参照ください。なお、`-a` オプションはファイルへの追記を意味しています。

複数のファイルやプロセスに渡せるということなので、こんなコマンドも実行可能です。ファイルをダウンロードして標準出力に投げて、`shasum` と `md5sum` でハッシュ値をとり、`dvd.iso` にダウンロードしたファイルを書き出し。

```
wget -O - http://example.com/dvd.iso \
| tee >(shasum > dvd.sha1) \
>(md5sum > dvd.md5) \
> dvd.iso
```

8.24 からエラーが起きた時、どの出力に書き出すか指定できるようになりました (`-output-error` オプション)。

^{*118} | や > はシェル組み込みです

^{*119} T の字形をみるとわかってくる

第 18 章

ファイル名の操作

ファイル名の操作をします。

18.1 basename

ファイル名からディレクトリや拡張子を取り除きます。

```
$ basename /usr/local/bin/sh  
sh
```

スクリプトの中で、`basename $0` と書くとそのスクリプト自身のファイル名が表示されます。ついでに、`basename $0 .sh` と書くと、.sh を除いたファイル名が表示されます。

18.2 dirname

ファイル名やディレクトリパスを引数にとり、ファイル名の最後の一部を取り去ります。実際には、ファイルパスの最後のスラッシュを取り去る挙動をします。ファイルがあるかどうかのチェックはしていません。

```
$ dirname /usr/local/bin/  
/usr/local/bin  
$ dirname /usr/local/bin/bash  
/usr/local/bin  
$ dirname /usr/local/bin/hoge  
/usr/local/bin # !?  
$ dirname /etc/etc/etc
```

```
/etc/etc # うーん
```

18.3 pathchk

ファイル名の SAN 値をチェックします。正確には、ファイル名を引数にとって、ファイル名をほかのシステムに持って行っても大丈夫かどうかチェックします。意訳すると、こんな感じです：

- パーミッションが関係でディレクトリの中身がみることができない
- ファイル名長すぎ

```
$ pathchk a<snip>a
pathchk: a<snip>a: File name too long
```

18.4 mktemp

一時的な空のファイルやディレクトリを作ります。bash スクリプトを書くときに、安全のため、使った方がいいけど、使わなくても何とかなります^{*120}。既存のファイルを重複しないファイル名やディレクトリを作ってくれます。X と書くと^{*121} その部分に適当な文字をあてがっててくれます。実際に使うときは、こんな感じです。作られたファイル名を取得します。

```
$ TMPFILE=$(mktemp hoge-XXXXXXX.txt) # この時点でファイルが作られます
$ echo $TMPFILE
hoge-82TiSmn.txt
```

その他オプションは下記の通り

- | | |
|---------------|--|
| -u XXX | XXX の長さだけ、ある程度ランダムな文字を表示します |
| -d XXX | ディレクトリを作ってくれます |
| -q | ランダム文字を表示します。ファイルは作られません。ランダムな文字列がほしい時に使うといいんじゃないでしょうか |
| -p dir | dir ディレクトリの下にファイルを作ります |

^{*120} 一時ファイルを作るときに mktemp を使っていると、ちゃんとしているなあという印象を与えることができます
[脳内調べ]

^{*121} X は 3 文字以上じゃないと怒られるので注意

18.5 realpath

相対パスやシンボリックリンクを絶対パスに直します。Coreutils 8.15 (2012-01-06) より加入。

```
$ realpath /tmp/../../tmp/../../tmp  
/tmp  
$ realpath hoge  
/home/ec2-user/hoge
```

第 19 章

わーきんぐの状況

働きましょう^{*122} ^{*123}。カレントディレクトリの状況を示します。

19.1 pwd

「ここはどこ？」コマンドです。今いるディレクトリを表示します^{*124}。もうこれ以上説明しなくちゃダメカナ？ダメダヨ？

オプションは下記 2 種類。

- L --logical 同じ。pwd のデフォルトの動作と覚えておけば不都合はなし
- P --physical 同じ。シンボリックリンクをたどる。つまりこんな感じ

```
[user@hostname]# ln -s /usr/local/apache2/logs /var/log/httpd # symlink  
[user@hostname]$ cd /var/log/httpd  
[user@hostname /var/log/httpd]$ pwd -L  
/var/log/httpd  
[user@hostname /var/log/httpd]$ pwd -P  
/usr/local/apache2/logs
```

^{*122} ちっちゃくないそうである。なお、この章のタイトルは Working Context。日本語訳は「作業環境」。良い翻訳だ

^{*123} 元ネタは高津カリノ先生による漫画作品「WORKING!!」。北海道某所のファミリーレストラン「ワグナリア」の従業員のドッタンバッタン大騒ぎを描く。アニメ化され 3 期まで制作された。ちっちゃくないとは背の小さい・・・じゃなかつた小柄な種島ばぶらの口癖である

^{*124} Print Working Directory

19.2 stty

端末のキャラクターを表示したり変更したりするコマンドです。端末とは、役所に設置されている住民票発行装置や、銀行の ATM を想像してみてください。とあるサーバの画面を離れたところにある画面に表示させているというイメージです。表示させて且つ操作することができます。パソコンに画面つなげて表示して操作するのも端末、ssh でリモートログインしているときも端末^{*125}。筆者はだいたいそんなイメージで端末という言葉を解釈しています。もっと年のいったおっさんが説明すると、もうちょっと古い端末を引っ張り出してきてボーとかテレタイプといった話をしだすと思います。

んでまあ、何が言いたいかというと、その端末の表示を変更するコマンドがこれなんです。原文を読んでみましょう。ライン設定が与えられていないとき、stty はボーレートを表示します。え？ マジで？

```
$ stty
speed 38400 baud; line = 0;
-brkint -imaxbel
```

baud(ボー) の登場です^{*126}。

中斷しちゃいました。続きです。stty はシステムがサポートしているライン制約ナンバー や stty sane で設定されている値から変更された設定を表示します。デフォルトでは、モードを読み込みや設定は、端末上の標準出力で実行されます^{*127}。-file オプションで変更可能ですが。stty は引数ではないたくさんの中身があります^{*128}。

主なオプションは下記です。

-a	-all と同じです。実行してみましょう
-----------	----------------------

```
$ stty -a
speed 38400 baud; rows 38; columns 79; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>;
eol2 = <undef>; swtch = <undef>; start = ^Q; stop = ^S; susp = ^Z;
rprnt = ^R; werase = ^W; lnext = ^V; flush = ^O; min = 1; time = 0;
-parenb -parodd cs8 -hupcl -cstopb cread -clocal -crtsets -cdtrdsr
```

^{*125} ssh だとエミュレータになりますけどね。端末エミュレーターでぐぐると深い話が垣間見れるのではないか。kterm とか xterm とか

^{*126} ボーは、搬送波に対する 1 秒間あたりの変調の回数と定義される (wikipedia より)。そして、1baud は 1bps と一致するかもしれないし、しないかもしれない。詳しくは wikipediaあたりを参照してください。

^{*127} よく考えなくとも当たり前というかなんというか

^{*128} 19.2.1 から 19.2.7 まで解説に割かれています

```
-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl ixon
-ixoff -iuclc -ixany -imaxbel -iutf8 opost -olcuc -ocrnl onlcr -onocr
-onlret -ofill -ofdel nl0 cr0 tab0 bs0 vt0 ff0 isig icanon iexten
echo echoe echok -echonl -noflsh -xcase -tostop -echoprt
echoctl echoke
```

- F device** -file=device と同じ。device に繋げます
- g** -save と同じ。ほかの stty につなぐため設定を保存しておくオプション

原文には、そのほかにオプションがたくさん載っているので、気になったら見てみましょう。

19.3 printenv

環境変数を表示します。それだけです。

19.4 tty

スタンダートインプット上のターミナルのファイル名を表示します。打ってみましょう。

```
$ tty
/dev/pts/0
```

上記の結果は、さくらの VPS サーバにログインして `tty` コマンドを打了結果です。さらに別の端末から同じサーバに入り、同じコマンドを打ってみましょう。

```
$ tty
/dev/pts/12
```

通常ならば、1になります。このサーバでは、`screen` が立ち上がっていて、1から11まで使っていました。つまり今回は、12に割り当てられました。詳しくはスペシャルデバイスファイルでぐぐってみましょう。

第 20 章

ユーザの情報

ユーザやグループなどの情報を表示します。

20.1 id

ユーザの id を表示します。打ってみましょう^{*129}。

```
$ id  
uid=10016(hoge) gid=20000(hpusers) groups=20000(hpusers)
```

筆者がこのコマンドを使うときは、あのユーザ wheel に入ってたっけ？と確認するときに使います。たとえばこのような感じ^{*130}

```
$ id ellen_jeager  
# 結果省略
```

なお、オプションは以下の通り

- | | |
|----|--|
| -g | -groupと同じ。グループIDを表示します |
| -G | -groupsと同じ。グループidと補助グループidを表示 |
| -n | -nameと同じ。ID番号じゃなくて名前で表示。-uか-g、または-Gが必須 |
| -r | -realと同じ。ID番号じゃなくてrealで表示。-uか-g、または-Gが必須。実行してみたけど、id番号しかでてこなかつ |

^{*129} value-server(レンタル共用サーバ)での結果。16人目のユーザなのかもねー。ちなみに、本当に「hoge」というユーザでユーザ名を作りました

^{*130} idのあとにユーザ名を複数入れられるようになりました(CoreUtils 8.31から)

たぜ

- u -user と同じ。ユーザ ID のみを表示
- z -context と同じ。セキュリティーコンテキストを表示。
SELinux が無効になっていたら警告を表示して return 1 する。

所属しているグループは、ログインした後に変更を加えても、今接続しているセッションには適用されないよ！ログインし直すと適用されるよ！と書いてあります^{*131}。

20.2 `logname`

現在のログイン名を表示します。`utmp` ファイル^{*132} から情報を読み出します。このファイルは、システムの現在の状態のすべてのアカウント情報を管理していて、システムが起動してからの時間^{*133} や、システムイベントやユーザのログイン、ログアウトの情報が記録されています。オプションは、`--help` と `--version` のみです^{*134}。

20.3 `whoami`

わたしはだあれ？コマンド。現在のユーザ `id` に関連づけられているユーザ名を表示します。同じコマンドは、`id -un`。

20.4 `groups`

所属しているグループ名を表示します。打ってみましょう。引数にはユーザ名を入れます。

```
$ groups hoge root
hoge : hpusers
root : root wheel rvm
```

`id -Gn` と同じコマンドです。

すべてのユーザ・グループを見るには、`getent group` します。

^{*131} かなり意訳

^{*132} `/var/run/utmp` か `/etc/utmp` にあります

^{*133} `uptime` コマンドを使って読み出します

^{*134} `--version` を打ってみたら、「Written by FIXME: unknown.」とでました

20.5 users

現在ログインしているユーザの名前を表示します。実行してみましょう。

```
$ users
fairy fairy
```

妖精さんだらけ——^{*135}。すべてのユーザ・グループを見るには、getent group します。

20.6 who

現在ログオンしているユーザの情報を表示します。コマンドの例は下記です。

```
who [option] [file] [am i]
```

原文をよく見ると、「am i」だけ斜体になってないんですよね。ということで実行してみましょう

```
$ who am i
chiba pts/11 2013-06-23 17:57 (:pts/12:S.10)
```

だいたい実行結果が見えてきたところで、オプションです。

- a -all や -b -d --login -p -r -t -T -u と同じです
- b -boot と同じです。システムが最後に起動した日時を表示します。uptime だと起動してからの時間が表示されます。意外と便利かも

```
$ who -b
system boot 2012-12-14 05:16
```

- H -heading と同じです。表示の際にヘッダをつけます
- l -login と同じです。訳すのが面倒だったので実際に打つところ感じです ^{*136}

^{*135} はみ出してるし。本当は自分の名字が並んでいただけでした。それだとおもしろくないので妖精さんを並べてみました。英語表記これであってるのかしら。無難に yo-say-san とかにしておいた方がよかったかも？生足魅惑のマーメイド？？つまり上半身は魚？？妖精さんの上半身は魚…ゞ ざわ…ざわ…

^{*136} 以前契約していた value-server で試しました。うすうす気づいていたのですが、これ、物理コンソールに root でログインしっぱなしになってないですかね

```
$ who -l
LOGIN    tty4          May 14 14:01      5500 id=4
LOGIN    tty2          May 14 14:01      5481 id=2
LOGIN    tty3          May 14 14:01      5491 id=3
LOGIN    tty1          May 14 14:01      5472 id=1
LOGIN    /dev/ttyS1    May 27 19:30      48033 id=v/tt
LOGIN    tty5          May 14 14:01      5503 id=5
LOGIN    tty6          May 14 14:01      5514 id=6
```

--lookup utemp に記録されているホスト名から DNS ルックアップしようとします。デフォルトでは機能しません。インターネットアクセスをするので表示するまでにちょっと時間がかかるからです

-m who am iと同じです

ここで、man コマンドを引きましょう。すると、

```
who [OPTION]... [ FILE | ARG1 ARG2 ]
```

コマンドの書式はこうなってますね。もうちょっと読み進めましょう。すると、

```
If ARG1 ARG2 given, -m presumed: 'am i' or 'mom likes' are usual.
```

だそうです。あとは察してください *137。

20.7 pinky

マニュアルに載っていません *138。でも man pinky すれば出てきます。なにをするコマンドかと言えば、finger の簡易版のコマンドです。finger って何ですか？良い質問ですね。ユーザの情報を探すプログラムです *139。実行してみましょう *140。

```
$ finger
Login      Name      Tty      Idle   Login Time   Office      Office Phone
root      root      pts/0      Jul 28 13:39 (hostname.example.com)
```

*137 3分くらい遊んで飽きる

*138 Coreutils のリポジトリに、TODO というファイルがありまして、その中の TODO の項目に入っています

*139 user information lookup program (man finger による)

*140 この実行結果に違和感を感じないのであれば、分かっていないか、分かってる人に分類できます

20.7. pinky

```
$
```

man の結果はこんな感じです(抜粋)。

SYNOPSIS

```
finger [-lmsp] [user ...] [user@host ...]
```

finger プロトコルを喋れるサーバに finger することができますが、もうそんなホストはないんじゃないでしょうか。 finger linux@kernel.org ってやると 最新のカーネル情報をとれるらしいんですが、もういないみたい^{*142}。 finger プロトコルについては、RFC1288^{*141} を参照。対して pinky はこんな感じ。 user@host がないですね。

SYNOPSIS

```
pinky [OPTION]... [USER]...
```

単に実行してみます。

```
$ pinky
```

Login	Name	TTY	Idle	When	Where
root	root	pts/0		Jul 28 13:39	hostname.example.com

pinky っていうのは、指に対しての小指という意味で名づけたのでしょう。おしまい。

^{*142} https://www.kernel.org/finger_banner っていうのがありますね

^{*141} <https://tools.ietf.org/html/rfc1288>

第 21 章

システムコンテキスト

システムの情報をえたり表示したりします。

21.1 date

役割は大きく分けて 2 種類あります。時刻を表示することと、時刻を設定することです。時刻の表示から説明すると思った？残念、時刻の設定でした！ *¹⁴³

時刻の設定。よく忘れます：

```
date [-u|--utc|--universal] [ MMDDhhmm[[CC]YY][.ss] ]
```

-u は使う機会はないでしょう *¹⁴⁴。6 月 25 日の 23 時 34 分 45 秒に設定したいときはこのように *¹⁴⁵

```
$ date 06252334.45
```

時刻の設定はこのへんにして、単独で実行してみます。

```
$ date  
Mon Jun 24 00:34:47 JST 2013
```

表示形式を変更したり、1 ヶ月後といった相対的な日付も出力することができます。オプション

*¹⁴³ 残念さやかちゃん

*¹⁴⁴ amazon aws だと引っかかりそうな。と思ったけど、日にちずれてることもないから、どうでもよかったです

*¹⁴⁵ 正確な時間がズレまくっていると、ntp でも合わせてくれないので、だいたい近日時に合わせましょう。ただし、時間を巻き戻す場合は、アプリケーションで不整合が起きることがあるので要注意。本番環境で気軽にやるなよ!!!

21.1. date

をいちいち説明するよりは実例を見たほうが早いです *¹⁴⁶

```
$ date +%Y%m%d  
20140727  
$ date '+%Y-%m-%d %H:%M:%S'  
2014-07-27 04:48:10  
$ date -d '1 hours ago' +%X  
23時20分22秒 # 日本語ロケールの場合こうなる
```

基本的な書き方はこれでマスターです。例えば、%Yと書くと今年の西暦である2014を表示、%yで西暦の下二桁、つまり、14を表示します。気をつけるところは、%が出現する一番最初に「+」の記号をつけること。さもないと、エラーになります。%に続く文字については、manを見ましょう *¹⁴⁷。次に、-d(または--date)オプションの説明をします。-dには、1時間前や1時間後、明日や先月などを指定することができます。「last month」と書けば、現在の時刻を基準とした先月の日時を表示することができます。適当に書いても結構柔軟に対応してくれます。ついでに指定できる文字列は、first,second,...,twelfth,last,this,first,next,tomorrow,yesterdayなどです。さて、この後は「21.1.7 Examples of date」を追っていきます。

date --date='2 days ago' おとといの日付を指定します

date --date='3 months 1 day' 3ヶ月と1日後の日付を指定します

date --date='25 Dec' +%j 1月1日から指定日までの日数を表示します

date '%B %-d' 月の完全名と月を表示します。%Bの部分はロケールによってJulyとか7月などに変わります。%-dの部分は、0埋めをしません。7月6日であれば、6となります。%dだけだと06になります

date --set='+2 minutes' 現在のシステムの時刻を2分進めます。root権限が必要です。システムの時刻を変更するときは注意だぞ！

date +%s 1970年1月1日からの秒を表示します *¹⁴⁸

date -d @946684800 1970年1月1日から経過した秒数を理解しやすい感じで表示してくれます *¹⁴⁹

ここで問題です。先月の最終日の日にちを出すワンライナーを作ってください。例えば、4月なら3月は31日まであるので、「31」を表示します。ここで答えを書いてしまうとすぐ見えてしまうので、この本の「おわりに」の脚注に載せました。

*¹⁴⁶ 完全に執筆してた時期ばれてますやん

*¹⁴⁷ %Y%m%d %H%M%Sあたりを覚えておけば、たいてい事足ります

*¹⁴⁸ 俗にいうエポックタイム。-dで日付を指定すればその時点のエポックタイムを表示します

*¹⁴⁹ Coreutils 5.3.0から機能です。これ以前だと、-d '1970-01-01 UTC 946684800 seconds' とする

date と crontab

crontab に date +%Y%m%d と書くと正しく動作しません。 % をエスケープしないといけないんだよ、な、なんだってーΩΩ筆者もハマったことがあります。 date +\%Y\%m\%d と書きましょう。これで動く！まあ、こうやって書くのが面倒ならスクリプトファイルにして、それを呼び出せばいいんですけど。

時刻合わせ

サーバの製造元が日本でない場合、製造した現地時間に合わせてあったりします。このとき、OS の iso イメージからマウントして、サーバに OS をインストールすると、BIOS の時刻を引っ張ってくるので時刻がズれます。そういえば、BIOS であわせるの忘れてた(・ω<)となります。仕方ないので最近はこのようにしています。当たり前ですけど、時刻がずれている場合は、OS をインストールしたらすぐ時刻をあわせましょう。

```
# ntpdate <ntp server name OR IP address> && \
> hwclock --systohc && hwclock --adjust
```

\$ cal 9 1752

date つながりで、カレンダーを表示する cal コマンドです。これは、ただのトリビアです。GNU には gcal という高機能なカレンダーコマンドがあります。詳細はこちらをご覧ください: <http://www.gnu.org/software/gcal/>

21.2 arch

uname -mと同じ。実行してみましょう^{*150}。

```
$ arch
x86_64
$ uname -m
x86_64
```

^{*150} \$ strace arch してみたら、uname 呼んでました

21.3 nproc

有効なプロセッサの数を表示します。実行してみましょう *¹⁵¹ *¹⁵²。

```
$ nproc
32
```

クールな例 *¹⁵³。

```
$ make -j $(nproc)
```

21.4 uname

現代語訳すると「coreutils の作りこみよ、我が前にシステム情報を表示せ、uname」になるんですかね *¹⁵⁴ *¹⁵⁵。実行してみましょう *¹⁵⁶。

```
$ uname -a
Linux e2.valueserver.jp 2.6.32-358.6.1.el6.x86_64
#1 SMP Tue Apr 23 19:29:00 UTC 2013 x86_64 x86_64 x86_64 GNU/Linux
```

オプションは下記の通り。

- a** すべての情報を表示。32bit か 64bit が分からなかつたらその部分は表示しないよ！
- i** ハードウェアのプラットフォームの名前 (x86_64 とか) を表示。カーネルによって情報が作られていないときは unknown って表示します
- m** ハードウェアの名前を表示 (x86_64 とか)
- n** ネットワークノードのホスト名を表示 (e2.valueserver.jp)

*¹⁵¹ 以前契約していた value-server での実行結果。こんなサーバなかなかお目にかかるないなあ。/proc/cpuinfo 見てみたら、本当に CPU が 32 個あった

*¹⁵² ついでに、このコマンドは coreutils 8.4あたりの比較的新しいバージョンに入っているみたいです。デフォルトの CentOS の 5あたりだと入っていないかも

*¹⁵³ <https://www.d7cb.de/blog/2010/nproc.html>

*¹⁵⁴ その前に、現代語訳じゃないでしょ。アニメ始まったの 1999 年だから 20 年くらい前だぜ。今思うと十分に中二病ですな、このセリフ

*¹⁵⁵ ↑「アニメ始まったの 1999 年だから 20 年くらい前だぜ」って書いてあったので修正 [第 7 版]。そりゃあ年もどるわな

*¹⁵⁶ 本当は 1 行で出力されます

-p	命令セットアーキテクチャとか ISA と呼ばれるプロセッサの型を表示。 unknown だったらオプションの -i と同じ理由
-o	OS の名前を表示 (GNU/Linux)
-r	カーネルリリースを表示 (2.6.32-358.6.1.el6.x86_64)
-v	カーネルバージョンを表示 (#1 SMP Tue Apr 23 19:29:00 UTC 2013)

21.5 hostname

コマンド単体で実行するとホスト名を表示します。ホスト名を変更する場合は、root ユーザで、引数に新しいホスト名を指定して実行します。一旦ログアウトしてログインすると、ホスト名が変更されています。再起動して元に戻ってしまうのを防ぐために、設定ファイルの変更も忘れずに *¹⁵⁸。あと、hostname を拾って起動しているプロセスにも注意を払っておきましょう *¹⁵⁷。

21.6 hostid

ホスト識別子を 16 進数で表示します。ソースを見ると分かるんですが、`gethostid` 関数を呼んでいます。試しに DigitalOcean のサーバでやってみるとこんな感じ。

```
$ hostid
00000000
```

なんでやねん。原文の例にはこうあります。

```
For example, here's what it prints on one system I use:
$ hostid
1bac013d
```

21.7 uptime

現在の時間とシステムが起動してからの時間と、ログインユーザの数と、ロードアベレージを表示します。ロードアベレージは、1,5,15 分の平均値です。`w` でも代用可能。

*¹⁵⁸ hosts ファイルも気にしてあげてください

*¹⁵⁷ mysql とか

21.7. uptime

```
$ uptime
11:44:16 up 49 days,  5:41,  1 user,  load average: 0.00, 0.00, 0.00
$ w
11:44:17 up 49 days,  5:41,  1 user,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM          LOGIN@    IDLE    JCPU   PCPU WHAT
user     pts/0    www5339u.sakura. Tue09    0.00s  0.56s  0.09s w
```

第 22 章

SELinux

SELinux(Security-Enhanced Linux) の設定を行います。root に権限が集中することを防ぐシステムの構築を提供します^{*159}。

22.1 chcon

選択されたファイルの SELinux セキュリティーコンテキストを変更します。

オプションは下記の通り

--dereference	シンボリックリンクに影響せず、シンボリックリンク先のファイルに影響します。デフォルトの動作です。
-h	シンボリックリンクのみに影響します
--reference=rfile	rfile と同じ設定になります
-R	再帰的な感じで動作します
-v	すべてのファイルについて調べて表示します
-u user	セキュリティーコンテキストをユーザ user に設定します
-r role	セキュリティーコンテキストをロール role に設定します
-t type	セキュリティーコンテキストをタイプ type に設定します
-l range	セキュリティーコンテキストをレンジ range に設定します

22.2 runcon

特殊な SELinux のコンテキスト上でコマンドを実行します。オプションです。

^{*159} wikipedia を参考にしました。ちゃんと知りたい方は調べてください (続きは英語で)

22.2. runcon

```
runcon context command [args]  
runcon [options] command [args]
```

-c 変更する前にプロセス変更コンテキストを計算する *¹⁶⁰

-u,-r,-t,-l chcon のオプションと同じ

*¹⁶⁰ Compute process transition context before modifying の訳。ムズイ

第 23 章

変更コマンド

いろいろ変更しまっせ。

23.1 chroot

特定のルートディレクトリでコマンドを実行します。スーパーユーザでのみ実行できます^{*161}。何がうれしいの？というと、本来ならば、/ディレクトリからのツリー構造になっているものを、任意のディレクトリ (/tmp/chroot/など) を/と再定義したツリー構造に変更することができます。つまり、本来の/は見えなくなります。ソフトウェアのテストなどを行うときに使います^{*162}。

実行はこんな感じ。

```
chroot option newroot [ command [args] ... ]  
chroot option
```

23.2 env

環境変数を表示したり、一時的に環境を変数を変更してコマンドを実行します。env 単体で実行すると、設定されている環境変数が表示されます。一時的に環境変数を変更するときの例。/tmp/bin というディレクトリに HOGE と出力される実行可能な hoge ファイルをおいておきます。

^{*161} 例外あり

^{*162} 原文に書いてある例を実行してみたんですがうまく動かず。本来だったら直下の必要なファイルを任意のディレクトリにコピーして実行するものらしい

23.3. nice

```
env PATH="$PATH:/tmp/bin" hoge  
HOGE
```

オプションは下記の通りです

-O,-null 出力時に改行しません *¹⁶³

-u name name という環境変数を削除します

-,-i,-ignore-environment 環境変数をすっからかんにして実行します。さっきのコマンドをアレンジしてみましょう。

```
env # まずは実行して様子見  
env - PATH="$PATH:/tmp/bin" env
```

なんとなく実行結果はつかめましたか？

23.3 nice

いいね！コマンドです *¹⁶⁴。niceness を変更してプログラムを実行します。niceness って何？いい質問ですね *¹⁶⁵。niceness は、システムで実行されるプロセスがいくつかあってその中でどれくらい有利に実行されるかを決める値です *¹⁶⁶。

単体で実行すると、現在の nice 値が表示されます。-20 (優先度高) から 19 (優先度低) までの値をとります。そして niceness はスケジュールの優先度と混同してはいけません。アプリケーションを実行する順序を決めます。niceness はスケジューラーに単に忠告するだけで、無視することもあります (そして原文では historical practice に脱線)。

実行例から。数値を与えればそれだけ nice 値が下がります。

```
$ nice  
0 # current nice  
$ nice nice  
10  
$ nice -n 19 nice  
19  
$ nice --10 nice
```

*¹⁶³ 新しめのバージョンに入っているオプションらしい

*¹⁶⁴ 正しくは like

*¹⁶⁵ いいね！（言いたいだけだろ!!）

*¹⁶⁶ 厳密には違うけど適当な説明ならこれでおっけー！（こらこら

```
nice: cannnot set nice ness: Permission denied
```

nice をつけてコマンドを実行すると、nice 値が 10 になります。 -n をつけると数値を指定できます。 - のあとに数値をつけると -n 数値 同じ効果です。 --10 はマイナス値にしようとしています。マイナス値をつけることは nice ness をあげることになります。 nice ness をマイナス値にするためには、root 権限が必要です。

```
$ sudo nice -n -20 nice  
-20
```

コマンドまとめて nice する場合はこんな感じです *¹⁶⁷。

```
$ nice -n 15 sh -c "command | command | command"
```

ちなみに、renice コマンドで実行中のプロセスの nice 値を変更することができます。多分いける。

23.4 nohup

ログアウトしても実行したコマンドを実行し続けることができるコマンドです。

```
$ nohup yes asumisu &  
$ logout
```

これで、いつでもイエッスアスミスし放題ですね *¹⁶⁸。実際に実行するときは注意してくださいね。

hungup シグナルを無視して、引数のコマンドを実行します。ログアウトしたあとでも引き続き実行されます。標準入力がターミナルのとき、/dev/null からリダイレクトされます。ターミナルのセッションはコマンドによって使われるターミナルと誤って見なさないようにするために *¹⁶⁹。これは GNU 拡張で、GNU でないシステムでは、nohup command [arg] ... </dev/null として下さい。

標準出力がターミナルのとき、コマンドの標準出力は nohup.out ファイルに追記されます *¹⁷⁰。

*¹⁶⁷ <http://unix.stackexchange.com/questions/22146/set-niceness-to-piped-command>

*¹⁶⁸ ディスクを食い尽くしてプロセスが落ちるところまでがオチです（ドヤア

*¹⁶⁹ これを書いている筆者もよくわかつてない

*¹⁷⁰ そのファイルに書けない場合は、\$HOME/nohup.out に書き、そこもだめだった場合はコマンドは実行されません。

nohup.out ファイルは、コマンドを実行しているユーザだけが書き込みができる権限で作られ

23.5. stdbuf

標準エラーがターミナルのとき、標準出力に出ます。標準出力が閉じられているときは、上記の nohup.out ファイルに追記されます。

例を見ましょう。

```
$ nohup make > make.log
```

ちゃんとバックグラウンドで実行するには、「イエッス、アスミス」で出てきたように、コマンドの最後に & を置きましょう。もし、 nice を使いたいときは、 nohup nice command としてください。

23.5 stdbuf

Coreutils 7.5 (2009-08-20) から追加されたコマンドです。説明が難しいので、原文直訳気味です。

i/o ストリームバッファリングを変更してコマンドを実行します。 stdbuf は、プログラムに関連づけられた 3 つの標準 I/O ストリーム (標準出力、標準入力、標準エラー出力) のバッファリング動作を 1 つに変更できます。オプションはもちろん 3 つ、

- i mode, --input=mode 標準入力ストリーミングバッファリングを調整します
- o mode, --output=mode 標準出力ストリーミングバッファリングを調整します
- e mode, --error=mode 標準エラーストリーミングバッファリングを調整します

mode に指定する文字は下記です

L ストリームを行単位でバッファします。このモードでは、新しい行が出力されるか、入力がターミナルデバイスに接続されたストリームから読まれるまで結合します。このオプションは標準入力では使えません。

0 大文字の o ではなく、ゼロです。選択されたストリームのバッファリングを無効にします。このモードでは、データはただちに出力され、要求されたデータの分だけが入力から読み込まれます。入力と出力のための機能の違いに注意してください。入力のバッファリングを無効にすると、応答性やストリーム入力機能のブロッキング動作に影響を与えません。基本的には、要求されたよりも少ないデータを読み取る場合でも、たとえば関数 fread の場合はまだ、EOF かエラーになるまでブロックします^{*171}。

size 完全バッファモードにおいて、バッファのサイズを明確にします。サイズは整数のあとに KB(1000 KiloBytes) とか K(1024KibiBytes) をつけることができます

ます。umask を無視します

*171 google 翻訳を借りました。以前見た時よりも翻訳が自然になっていて驚き。そして翻訳された日本語の文章の意味はイマイチ把握できず

```
tail -f access.log | stdbuf -oL cut -d ' ' -f1 | uniq
```

このコマンドでは access.log の一意なエントリがあると直ちに output されます^{*172}。

```
command | stdbuf -oL gawk \
'{print strftime("[%a %b %e %H:%M:%S %Z %Y] "), $0; }'
```

コマンドの標準出力にタイムスタンプを追加する例です^{*173}。このコマンドの具体的な活用例はあまりないので、エントリを上げるチャンスですよ！

23.6 timeout

タイムリミットを設定してコマンドを実行します。Coreutils 7.0 (2008-10-05) より加入。CentOS 5 系だと入っていないかも。

```
timeout [option] duration command [arg] ...
```

オプションは下記の通り

--preserve-status タイムアウトを示す具体的な終了ステータスを返します。どのくらいかかるかわからないコマンドを実行するときに便利
*174

--foreground 正常フォアグラウンド TTY を使用できるように、独立したバックグラウンドプログラムループを作成しません。これは、コマンドが 2 つの状況で、対話型シェルから直接起動していないコマンドがタイムアウトをサポートするために必要とされます。2 つの状況とは、1. command がインタラクティブで、例えば端末から読み取る必要がある場合 2. 端末からコマンドを直接送信したい場合、たとえば Ctrl-C など

-k duration, --kill-after=duration 監視コマンドは、指定した期間のあとに、KILL シグナルを送ることによって kill されていることを確認しま

*172 日本語マニュアルからお借りしました

*173 <http://qiita.com/yyamamot/items/60a0a007b0016da61b32>

*174 原文にはスペルミスがあるので訳すときは注意。と、書いていたのですが、この本を読んでいただいた方から、Coreutils にバグレポートを送っていただきました。最新の Coreutils のマニュアルでは修正済みです。やりとりはこんな感じでした (https://twitter.com/okano_t/status/507278269524082689)。@okano_t さん、ありがとうございました。バグレポートってこんなかんじで送るのかーと知見を得ました

23.6. timeout

す。選択された信号が致命的でないと証明された場合、このオプションを指定しない場合、コマンドを kill することはありません

-s signal, --signal=signal デフォルトである TERM シグナルではなく、タイムアウトの時に signal をコマンドに送ります。signal は HUP や数値で指定します。単に数値の場合は秒です

期間に関しては、小数の後に、s(秒、デフォルト)、分である m、時間 h、日 d をとることができます。期間が 0 の場合、タイムアウトになりません。実質のタイムアウトの期間はシステムの状態に依存します。特に注意しないといけないのは、1 秒以内のタイムアウトです。

「イエス、アスマス」に細工します *176。

```
$ timeout 5 nohup yes asumisu &
```

5 秒後に実行が終わるので、ディスクにも安心。

*176 この言い方が適切かどうかは不明

第 24 章

プロセスコントロール

この章には kill しかありません。

24.1 kill

非実在の妖精さん(プロセス)にお菓子(シグナル)を与えます^{*178}。お菓子(シグナル)にもいろいろあって、それを食べた妖精さん(プロセス)は、消えてみたり、いっぺん寝て起きてみたり、「どうされましたか?」^{*177}などと言ってみたりします。お菓子によってどのような行動をするかは、妖精さんごとに定義されています。

kill コマンドのデフォルトでは、TERM というお菓子を妖精さんに与えます^{*181}。お菓子(シグナル)には、番号や名前がついています。HUP(番号だと 1)^{*179}、KILL(番号だと 9)^{*180}などです。詳細は、Coreutils のマニュアル、2.5 Signal specifications に載っています^{*182}。なお、ちゃんと言うことをきいてくれる(アクセス権限のある)妖精さんにしか効果ありませんのでご注意を。

よくあるコマンドを示します。「-9」というお菓子を、妖精さんの番号「12345」(プロセス ID/PID)に与えます。2つのコマンドとも同じ意味です。

```
$ kill -9 12345
$ kill -KILL 12345
```

*178 ググってみるとプロセスを強制終了するコマンドという説明を見かけます。これは、正確ではありません

*177 null signal のイメージ(多分ちょっと違う)

*181 システムコールの世界によるこそ

*179 ログをローテートさせるときに使います。apache の logrotate であれば

*180 プロセスを終了させるときに使います。通常「殺す」って言います。なお、このシグナルは、受け取り拒否または無視できません

*182 あのシグナルなんだっけ?というときは、\$ man 7 signal してください

24.1. kill

妖精さんの番号が正の数の時、0 の時、-1 の時、-1 より小さい時の説明があります。気になる人は自分で調べてみてください。なお、killall というコマンドがあります。これは、引数にプロセス名を取ります。root で単に killall を実行すると、すべてのプロセスが終了します (UNIX System V バージョンの場合) ^{*183}。

^{*183} このへんは wikipediaあたりを参照しました

第 25 章

遅延

原文は Delaying です^{*184 *185}。

25.1 sleep

指定された時間だけ実行が遅延します。アラームとか実行するといいんじゃないでしょうか。

```
$ sleep 1d 1h 1m 1s && echo $'\a'
```

1 日 +1 時間 +1 分 +1 秒後にビープ音を鳴らします^{*186}。もちろん、このコマンドを実行するマシンは足元にないとだめですよ^{*187 *188}。

^{*184} 本当は「私に遅延が舞い降りた!」ってタイトルにしようと思ってたんですけど、遅延が舞い降りるってなんだよ? と思って止めました。元ネタは「私に天使が舞い降りた!」

^{*185} 「私に天使が舞い降りた!」は、女子小学生と仲良くなりたい女子大学生がお菓子を焼いたり世話を焼いたりするアニメーション作品。アニメの作画が神。KAWAII アトモスフィアが襲ってくる。漫画原作ですがアニメになったら化けたという印象 [脳内調べ]

^{*186} 筆者の部屋に転がってた FreeBSD の入った実機で echo \$'a' を実行してみたら「ピッ」と鳴った。手元の MacBookAir のターミナルでやってみても鳴った

^{*187} クラウド上のサーバがピーーなっちゃいますよ (ホントかなあ

^{*188} 以前契約していた value-server で大量に実行してみたけど怒られなかったし、いいんじゃないでしょうか。そもそも一般ユーザで鳴るのかね? まあいいか

第 26 章

数値操作

26.1 factor

「せんせい！そいんすうぶんかいがしたいです」「よろしい、ならば戦争（ry）」「おいやめろ！！」
ということがないように、素因数分解ができるコマンドがあります。

```
$ factor 60  
60: 2 2 3 5
```

原文だと唐突に、メルセンヌ素数をもとに実行例が出てきます。曰く、8番目と9番目のメルセンヌ素数^{*189}を計算するときには、Athlon の 2.2GHz の CPU で 30 ミリ秒くらいかかります^{*190}、とあります。

```
M8=$(echo 2^31-1|bc)  
M9=$(echo 2^61-1|bc)  
n=$((M8 * $M9) | bc)  
/usr/bin/time -f %U factor $n  
4951760154835678088235319297: 2147483647 2305843009213693951  
0.03
```

さらに読んでいきましょう。8番目のフェルマー数^{*191} ($2^{256}+1$) は 20 秒くらいかかります^{*192}。

^{*189} 2^n-1 (nは自然数)の形の自然数且つ素数。ちなみに 48 番目が発見された日は、2013 年 1 月、って最近じゃないか。物好きな方は GIMPS でググると吉

^{*190} さくらのサーバでやってみたら桁数多すぎて怒られました。value-server でやってみたら、0.02 でした

^{*191} フェルマー数とは、 $2^2 \cdot 2^n + 1$ (nは自然数)

^{*192} この文章書かれたのいつなんでしょうね？

大きい数になると一般的に求めるのが難しくなります。比較的小さい数字を求めるときに、ボラード・ロー因数分解法が使われます。でかい数で且つ素因数がおっきい数を求めるときは、もっとほかの方法をとった方がよさげです。

GNU MP^{*193} を使わずにビルトされた factor コマンドは、single-precision 算術が有効になります。その算術方法は、小さい数字を計算することが得意で、2^64 以上の数字はサポートしていません。158909489063877810457 や 222087527029934481871 の素因数分解をするときにループに陥りましたが、coreutils 8.20 でバグってて、8.27 で修正されました。普通こんなことしないと思いますが念のため。

「せんせい！ 1000 までの素数が知りたいです」「よろしい、ならば戦争（ry）と、なってしまったら？ここで答えを書いてしまうと面白くないので、あとがきに載せました。

26.2 numfmt

Coreutils 8.21 (2013-02-14) から使える比較的新しいコマンドです^{*194}。例えば、4G を 4,000,000 に変換してくれます。例をいくつか拾ってみましょう。詳しいことはマニュアルを読みましょう。

```
$ numfmt --from=auto 1Mi
1048576

$ numfmt --to=si 500000
500K

# Third field (file size) will be shown in SI representation
$ ls -log | numfmt --field 3 --header --to=si | head -n4
-rw-r--r-- 1      94K Aug 23 2011 ABOUT-NLS
-rw-r--r-- 1     3.7K Jan  7 16:15 AUTHORS
-rw-r--r-- 1     36K Jun  1 2011 COPYING
-rw-r--r-- 1        0 Jan  7 15:15 ChangeLog

$ LC_ALL=en_US.utf8 numfmt --from=iec --grouping 2G
2,147,483,648
```

^{*193} GMP といって、多倍長演算ライブラリのことです。http://gmplib.org を参照

^{*194} ついでに、4 章 (fmt とかがあるところ) からこの章に説明が移動してました

26.3 seq

連続した数字を表示します。使う頻度はそこそこ多いかも知れないです^{*195}。実行例はこんな感じ。

```
$ seq 3
1
2
3
$ seq -w 7 10
07
08
09
10
$ seq 1 2 5
1
3
5
$ seq -w 10 -2 2
10
08
06
04
02
$ seq -s "," 10 -2 2
10,8,6,4,2
$ seq -s + 1 100 | bc
5050
```

オプションに関しては察してください。あとは-f オプションで printf と同じようなフォーマットが使えるモードがあります。あとは非常に大きい数字の場合、5e+06 といったような表現になるときがあります。そんなときは、-f を使ってフォーマットを指定して解決。

```
$ seq 20140401 20140410
2.01404e+07
2.01404e+07
2.01404e+07
2.01404e+07
```

^{*195} bash のカッコ展開で代用することも多いかも。{1..100}とか

```
2.01404e+07
2.01404e+07
2.01404e+07
2.01404e+07
2.01404e+07
2.01404e+07
$ seq -f %1.f 20140401 20140410
20140401
20140402
20140403
20140404
20140405
20140406
20140407
20140408
20140409
20140410
```

seq 1 0 10 ってコマンドを実行すると無限ループしていました。インクリメントが0ですね。無限ループ回避として、coreutils 8.26 からエラーになります。見逃されていたというかなんというか。

第 27 章

答え合わせ

文中で出てきた問題の解答例と解説です。脚注の [練習問題] は、自習ということでひとつ。

27.1 date

date コマンドで出てきた問題の解答です。前月の最終日の日付を表示するワンライナーでしたね。こちらです。

```
date -d $(date +%Y%m01) '-1day' +%Y%m%d
```

計算しててずるい感じしますがこうするしかなかったです。もっと短くかける方、いますぐ筆者までリプライください。

27.2 factor

factor で出てきた、1000までの素数を表示するワンライナーです。某所でバズったのでご存知の方がいるかもしれません^{*196}。解答例はこちらです。

```
$ seq 1 1000 | factor | awk 'NF==2{print $2}'
```

解説すると、seq で 1 から 1000 までの数値を出して factor に食わせます。

^{*196} 元ネタはこのへんです <http://oki2a24.com/2014/03/03/how-to-print-prime-number-to-10000-with-shell/>
<https://twitter.com/usptomo/status/479858878310383616>

```
seq 1 1000 | factor | tail
991: 991
992: 2 2 2 2 31
993: 3 331
994: 2 7 71
995: 5 199
996: 2 2 3 83
997: 997
998: 2 499
999: 3 3 3 37
1000: 2 2 2 5 5 5
```

次に awk を使ってフィールドの数が 2 つのものだけを抽出します。フィールドというのは、例えば「999: 3 3 3 37」でいうことの「999:」「3」「37」にあたる部分です。この場合、フィールド数は 5 です。フィールドが 2 のものは「991: 991」とか「997: 997」とかで、素数になっていますね。あとは、2 番目の文字を出力 (print \$2) すれば終わりです。

さて、sed 派の皆さん、お待たせしました。 sed で処理をするならこちらです^{*197}。解説をすると、

```
seq 1 1000 | factor | sed -n '/: [^ ]*$/ {s/.*/: //; p}'
```

オプションの解説です^{*197}。

-n オプション p コマンド (print) が指定されない限り勝手に表示しない
`/: [^]*\$/` AWK でいうパターンに相当するもので、`: ``の後ろにスペース無し (つまり素数行) だけ反応する

{~} 上記のパターンにマッチしたら括弧の中のコマンドを実行

s/.*/: // 1 列目の文字列を消す

p その結果を表示する

^{*197} sed 方式は @richmikan 氏に解説含め教えていただきました

第 28 章

おわりに

ここまで読んでいただきありがとうございました。脱線しまくりの「解説 Coreutils」いかがだったでしょうか。原文をまじめに読むと、挫折します。実際に挫折しかけました。ただ、通読しておくと、こんなことができるという印象だけ残って、いざというときに、アレが使えるというのを思い出して状況を打破できることがあります。多くを知っておきましょう。損はしません。

最後に、このコマンドを俺が一番うまく使えるんだ！という Tips をお持ちの方、この環境だとこの辺でこけるといった検証報告をお持ちの方、この説明違うよ！全然違うよ！！ということを思われた方は、筆者^{*5}まで連絡を頂けると大変ありがとうございます。第2版が出るその日までさようなら。

Let's enjoy coreutils life. *1

28.1 第 2 版おわりに

早くも、第2版がでてしまいました。思ったより反響が大きくて涙ちょちょぎれてげっそりしています。先日、健康診断の結果が返ってきました。体重は安定していました。なお、中性脂肪は前回の3倍くらいになりました。もっとラーメンを食べたいと思いました（コナミ

前回からの変更点としては、索引がつきました！！パチパチ！！！初版を使っていて、索引がないことにイライラしていました。私のtwitter上で索引をつけていただける方を募集したところ、Pull req いただきました。shirou^{*7}さんありがとうございます！

そして、この原稿の全文(表紙とあとがき除く)は <https://github.com/nanaka-inside/kaisetsu-CoreUtils> にあります。Pull req歓迎しております。そのときは第3版を出してしまいかもしれません。皆様、よろしくお願ひ致します。

^{*5} [連絡先] <https://twitter.com/tboffice> または、tbofficed@gmail.comまで

^{*1} ここで書くのも何ですけど、<http://www.commandlinefu.com/> が便利

^{*7} https://twitter.com/r_rudi

28.2 第3版おわりに

ティアズマガジン 109^{*10} に、本誌が掲載されてしまいました。推薦していただいた方、ありがとうございます。そのおかげか、第2版の在庫がなくなり、こうして第3版を出すことになりました。変更点としては、Coreutils のマニュアルの timeout コマンドのところでタイプがあることを指摘していました。それを Coreutils のバグレポートに送って頂いた方がいらっしゃいました、それが反映されましたということを追記しました。なお、Coreutils - rejected feature requests^{*11} というページがありまして、これを訳したかったんですが時間がなく。あとは、コマンドの結果を標準出力とエラー出力にわけて欲しいという要望もあったのでなんとかしたいところ。もちろん時事ネタとか、とりとめもなく雑多に書いてしまった部分があったのでその辺も整理したら第4版が出るのかもしれません。話は変わりますが、私の職場の後輩にも、コマンドの勉強しろと言って、この本を渡しています^{*12}。教育機関や企業からの受注をお待ちしております^{*13}。

28.3 第4版おわりに

Final です^{*14}。毎回 100 部ずつ印刷してるし、世の中の必要な人には行き渡ったかなーとおもいきや毎回完売してしまうのが非常に不思議でなりません。今回は奮発してちょっと多めに作ったのでしばらくは印刷しません。ここまでお読みいただきありがとうございました。@tboffice 先生の次回作にご期待ください^{*15}。

末筆ですが、coreutils の別実装があるので紹介します。

go 実装 <https://github.com/EricLagergren/go-coreutils> クロスプラットフォーム実装です。完成間

近！？

Rust 実装 <https://github.com/uutils/coreutils>

28.4 第5版おわりに

一年経ちました。在庫がなくなりました。前回 Final つってんのに第5版ですよ。何なんですか（逆ギレ

今回は編集・ビルト・入稿まで 2 日でやるとか頭おかしい感じでした。一年ぶりくらいにビルトしようとすると、環境がなくなっていたりしてよろしくないですね。今回はそれを見

^{*10} 創作オンラインの即売会・コミティアのカタログのこと

^{*11} https://www.gnu.org/software/coreutils/rejected_requests.html

^{*12} ひどい先輩がいたもんだ #おい

^{*13} おい

^{*14} 多分

^{*15} おい！！きいてねーぞ！えーと、次は gnu findutils とか書けばいいんですかね？？？(乗り気)

28.5. 第 6 版おわりに

越して sphinx+latex をビルドするための docker イメージを作っていました^{*16}。sphinx1.3.6 + TeXLive2015 でビルドしました。docker イメージのサイズがでかいので、改良しようとしている間に TeXLive! サンシャイン!!じゃなかった TexLive 2016 が出来てしまいアチャーという顔をしております。

さて前回との比較(実質的には、Coreutils 8.24 と 8.25)をすると、base32 コマンドが追加されました。マニュアルベースの変更では、sort -R に shuf コマンドの説明が追加されたり、groups, users コマンドに似たコマンドとして getent, who コマンドの記述が追加されたりしたくらいです。細かいオプションの変更やらなんやらありますけど、些細すぎるので割愛です。

次回があるなら、Coreutils - rejected feature requests をおまけとして載っけたいんですが間に合うか、というところ。本書、第 1 章が 2 つあるのは見なったことにして下さい。ではまた～

28.5 第 6 版おわりに

第 5 版は、アキバ・スクエアで開催された技術書典 2 で売り切れてました。あのイベントはめっちゃ売れた。ビビるほど卖れた。コミケより売れたんじゃないかな。怖いイベントでした。ということで、今回は第 6 版です。第六版と書くと広辞苑ぽいですね。追いつきましたね。そうじゃない。

最近は日本酒ばかり飲んでます。日本酒度が +2 以下を目安にしています。気づいたら、お腹が出てきました。つい先日、健康診断の結果が返ってきました。三ヶ月後再検査してね、肝機能に軽度の障害があるよ、だそうです。さもありなん。

今回こそ「Coreutils - rejected feature requests」^{*17}を翻訳してみたかったんですが、この本の趣旨じゃないよなーってことでやめました。なにこれ? という人に解説しますと、過去にツッコミのあったお節介集ですね。言い方に気をつけないと刺されますね。あぶないですね。結構量がありまして、別冊で薄い本ができる流れです。完全にフラグです。雑に github に意訳を上げるだけでいいかな? ダメかな?

さて、ここからは恒例のアップデートコーナーです。前回と比較(8.25 と 8.27)すると、b2sum コマンドが追加されました。factor がループに入るバグとか、seq で無限ループするとかが直っています。今回も docker のイメージがありまして助かりました。久々に引っ張り出してビルドしたんですが、一発で PDF が生成されて、すげーなー(白目)ってなりました。動作する環境って重要ですよね(涙目)。ではこのへんで。

^{*16} Sphinx+TeXLive2015+jenkins の docker イメージを作る話 <http://qiita.com/tboffice/items/6004c9f99bf4dbc2d0ff>

^{*17} https://www.gnu.org/software/coreutils/rejected_requests.html

28.6 第 7 版おわりに

第 6 版を印刷してから一年半が過ぎました。こんにちは筆者です。三十路も半ばとなり話題は健康の話に花が咲きます。桜の花も咲いている時期でしょうか^{*18}。このところ、1 年ほどで在庫が払底し、次の版を出すペースで来ておりましたが今回は遅くなりました。理由は印刷費がなかったのです！今この本の印刷費が出来るかどうかの瀬戸際です。いつもの印刷所なので交渉しています。そういうこともあります。こんなことは 2 回目です。おい！

さてアップデートのコーナーです。2019 年 3 月 10 日に 8.31 が出ました。これを書いている二週間くらい前でした。8.30 からの主な修正点は、

- basenc コマンドが追加。この版からコマンドを追加
- base64 hoge fuga したときに hoge をオプションとして解釈していたのを修正
- ファイル B が存在する状態で cp -il A B した場合に失敗するのを修正
- macOS で df コマンド実行時、マルチバイト文字がある場合に表示が崩れるのを修正
- sort /dev/null -o /dev/stdout | cat が実行できるようになった
- test -a FILE はもはや機能しなくなった。test -e FILE を使う
- date コマンドは変換指定フラグ + をサポートした。date --date=12019-02-25 +%+13F とすると +012019-02-25 と出力される。POSIX.1-2017 だそうである
- test コマンドは新しく -N FILE をサポートした！
- stat と tail コマンドはアンドロイド上の sdcardfs を認識するようになった
- stat コマンドは、GNU Linux システムの glibc >=2.28 とカーネル >= 4.11 でサポートされているファイルの作成日時を表示する

今更こんなバグが！ってのもありますね。Coreutils は日々進化中！^{*19 *20 *21}。

さてこの原稿をビルドするために前回と同じく docker image を引っ張りだしてきました。初心者にありがちな「何もしてないのに動かない」という状態になりました。それもそのはず、docker image の Git クライアントのバージョンが 1.2 だったため github が受け付けず、また https のプロトコルもバージョンが古くエラーになりました。OS のバージョンを上げて再ビルドしなきゃいけないですね。やっぱりメンテ（テスト）は必要ですね。

^{*18} 第 7 版を出したのは 4 月の上旬

^{*19} レビュースタライターの愛城華恋のセリフのように言ってみたけど伝わらない気がする。元ネタは彼女のセリフにある「愛城華恋は日々進化中」から

^{*20} レビュースタライターは「ミュージカル×アニメーションで紡ぐ、二層展開式少女歌劇」。アニメーションもやるしミュージカルもやるという手の込んだ作品。スマホアプリもあるよ！

^{*21} 筆者の推しはフロンティア芸術学校の野々宮ララフィンです

索引

-, 3
-, 3
[, 56

arch, 71

b2sum, 17
base32, 8
base64, 8
basename, 58
basenc, 8

cat, 5
chcon, 75
chgrp, 46
chmod, 47
chown, 46
chroot, 77
cksum, 17
comm, 23
cp, 36
csplit, 14
cut, 26

date, 69
dd, 38
df, 48
dir, 35
dircolors, 35
dirname, 58
du, 49

echo, 51
env, 77
expand, 29
expr, 56

factor, 86
false, 53
fmt, 10
fold, 11

groups, 65

head, 12
help, 2

hostid, 73
hostname, 73

id, 64
install, 38

join, 27

kill, 83

link, 42
ln, 42
logname, 65
ls, 31

md5sum, 17
mkdir, 43
mkfifo, 43
mknod, 43
mktemp, 59
mv, 39

nice, 78
nl, 7
nohup, 79
nproc, 71
numfmt, 87

od, 8

paste, 27
patchchk, 59
pinky, 67
pr, 10
printenv, 63
printf, 51
ptx, 24
pwd, 61

readlink, 44
realpath, 59
rm, 39
rmdir, 44
runcon, 75

seq, 87

sha 系, 18
shred, 41
shuf, 21
sleep, 85
sort, 19
split, 13
stat, 49
stdbuf, 80
stty, 61
sum, 16
sync, 49

tac, 6
tail, 12
tee, 57
test, 54
timeout, 81
touch, 47
tr, 29
true, 53
truncate, 50
tsort, 24
tty, 63

uname, 72
unexpand, 29
uniq, 23
unlink, 44
uptime, 73
users, 65

vdir, 35
version, 2

wc, 16
who, 66
whoami, 65

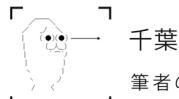
yes, 51

解説Coreutils

カイセツコアユーティルズ

2014年05月05日 初版 発行
2014年08月17日 第2版 発行
2014年12月30日 第3版 発行
2015年08月16日 第4版 発行
2016年08月14日 第5版 発行
2017年08月11日 第6版 発行
2019年04月14日 第7版 発行

編集後記



千葉

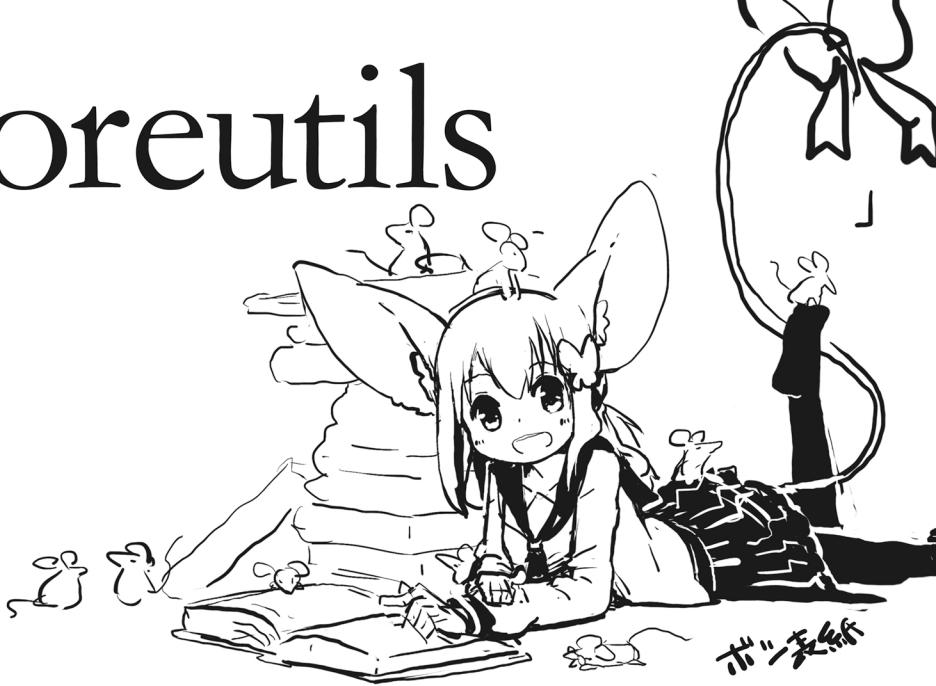
筆者の "@tboffice" こと千葉です。
本家から怒られたりしないかどうかビクビクしています。Bashな本はあるんですが、Coreutilsに絞った本ないんですよね。1年くらい前にこの企画を思いつきまして、いよいよ単行本で出ることになったかと感慨深いです。表紙や本文の一部を、以前にお手伝い頂いた矢上さんにお願いしました。ステキな表紙が上がって来て涙ちょちょぎれました。ありがとうございます。また、本文の体裁のサポートをしてくれた@mtgto氏と、妖精さん描いてよ!って無茶ぶりしてしまいました@saiten氏にお礼申し上げます。カバーフルと一気に単価が跳ね上がりますね(白目)



矢上

という訳でCoreutils本お疲れ様でした
"ななかVol.3"よりお声頂きまして、今作もデザイン協力としてお手伝いさせて頂きました
矢上諒一です。

千葉さん命名により上を向いてきゅうりの漬物を食う現代に蘇る人型ネズミ娘が誕生しました。最初は臆病なキャラで描いていたハズだったんですが何故かメス5匹の女主体家族(?)の長女に生まれてしまった彼女は、どうやらそんな軟弱な事では兄弟を守っていけない!と思い込んだのか勝手に活発なキャラへと変貌してしまいました、キャラの1歩きつてするんですね(投げやり)。ともあれそんな"きゅー"がマスクコットのCoreutils本、楽しんで製作に参加させて頂きましたありがとうございます。また機会があれば今度は42km位勝手にダッシュしてくれそうなキャラでも作りたいと思います、ではでは。



企画・製作

第7開発セクション

千葉 / @tboffice

Mtgto / @mtgto

Saiten / @saiten

表紙デザイン & キャラクターデザイン

矢上 諒一 / Ryo Yagami [Ye*]

Web: <http://yew.digiweb.jp/>

発行

第7開発セクション

Web: <https://sites.google.com/site/dai7sec/>

Mail: dai7section@gmail.com

印刷会社

株式会社 イニュニック

Coreutils

—おーい磯野ー、シェルスクリプト書こうぜー

技術系同人サークルの第7開発セクションが発行した
"ななかInside PRESS"に連載されたCoreutilsの
マニュアルを筆者の経験と実例を交えて書き直し
てみました。

マニュアルをきちんと読むと、あのコマンドに
こんなオプションがあったの？！このコマンド何に
使うの？！といった発見があります。コマンドの
オプションも一通り解説している為、こんな機能が
あるんだということを覚えておくだけでレベル
アップ！

どこかで見たことがある本と同じサイズ、同じ
ような用紙で製本したので、それらの本棚にそっと
しおばせておけばカモフラージュは完璧です。



発行元／第7開発セクション
