

```

4780 GOTO 5000
4790 :
4800 REM -----
4801 REM --- DARSTELLUNG ---
4802 REM --- DES MANUALS ---
4803 REM -----
4810 :
4820 PRINT " ";
4825 W=V+1:IF W<0 THEN W=W+14
4830 FOR X=1 TO 2:PRINT " ";
4835 FOR I=0 TO 23
4840 PRINT MD$(I+W);
4850 NEXT:PRINT:NEXT
4860 PRINT " ";
4870 FOR I=0 TO 23
4880 IF MD$(I+W)=CHR$(32) THEN PRINT MB$(I+1);:GOTO 4900
4890 PRINT MD$(I+W);
4900 NEXT
4910 PRINT:PRINT " ";
4920 FOR I=2 TO 24 STEP 2
4925 PRINT "|";
4930 IF MD$(I+W-1)=" " THEN PRINT " ";:GOTO 4940
4935 PRINT " ";
4940 NEXT:PRINT " "
4950 PRINT " ";
4960 FOR I=2 TO 24 STEP 2
4965 PRINT "|";
4970 IF MD$(I+W-1)=" " THEN PRINT " ";
4975 PRINT MB$(I);:GOTO 4980
4980 NEXT:PRINT " "

```

基本ソフトウェア

(OS-オペレーティングシステム)

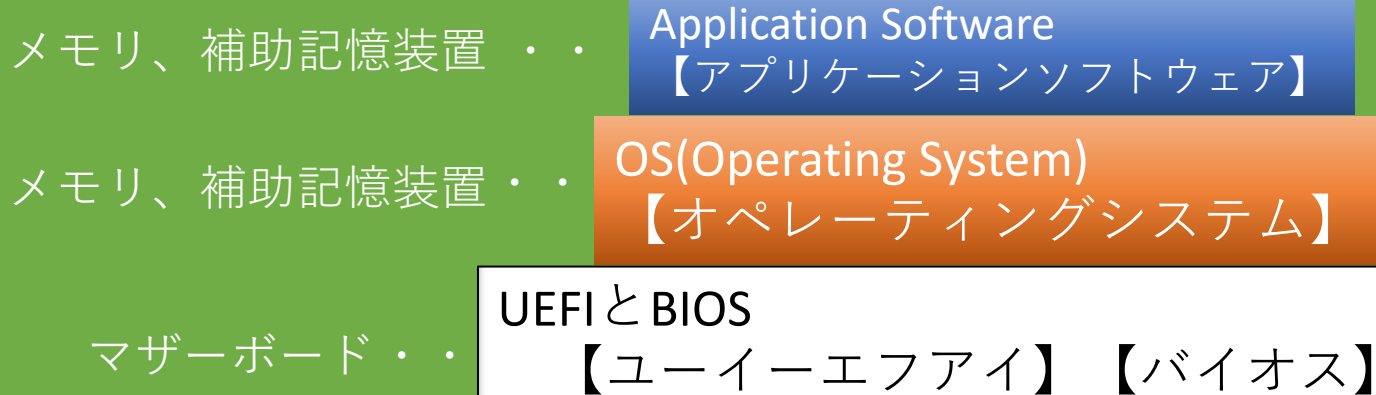
基本情報技術者 第5回

Roots千葉 利用者

長岡 昇吾

OSの仕事

ハードウェア内の帰属



OS(オペレーティングシステム)の種類

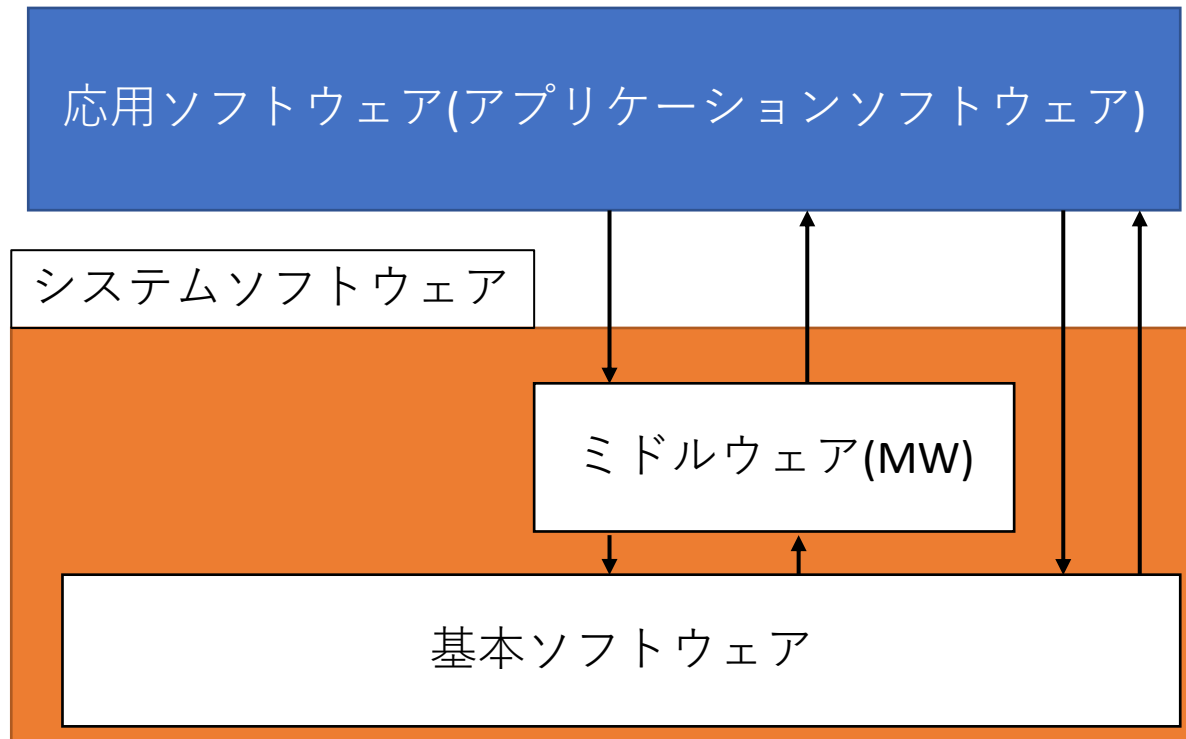
1.PC系

- Unix
- Mac(Unixベース)
- Linux
- Chromium(Linuxベース)
- Windows

2.スマートフォン

- iOS
- Android

OSの仕事



- | | |
|---------------|----------------------------------------------|
| 1.基本ソフトウェア・・・ | OSなど基本的な機能を提供するソフトウェア |
| 2.ミドルウェア・・・ | GUIやデータベース管理システム(DBMS)などソフトウェアから利用される機能を提供する |
| 3.応用ソフトウェア・・・ | ExcelやWord、Acrobatなど実際に私達の使う部分 |

OSの仕事

基本ソフトウェア



制御プログラム(カーネル)

プログラムの実行状態を管理したり
ハードウェア資源を管理してアプリケーションソフトが
ハードウェアの機能を利用する手段を提供したりする。

言語処理プログラム(言語プロセッサ)

Java→C言語→アセンブラ→機械語
のようにコンピュータが理解できる言語に翻訳する。

サービスプログラム

コンピュータの機能を補う、補助的なプログラムのこと
で、ユーティリティとも呼ばれます。たとえばファイル
圧縮プログラムなどが該当します。

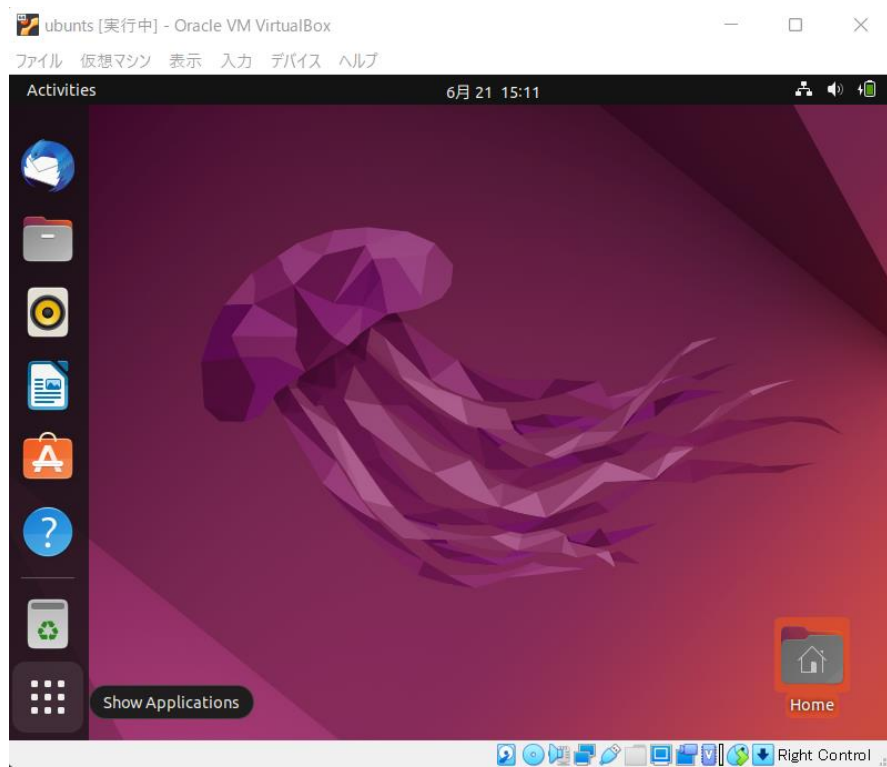
実はOSにはGUI(グラフィックユーザーインターフェイス)とCUI(キャラクタユーザーインターフェイス)などがあります。

GUIはMacやWindows、Chromiumなど視覚的に分かりやすくなっているもの

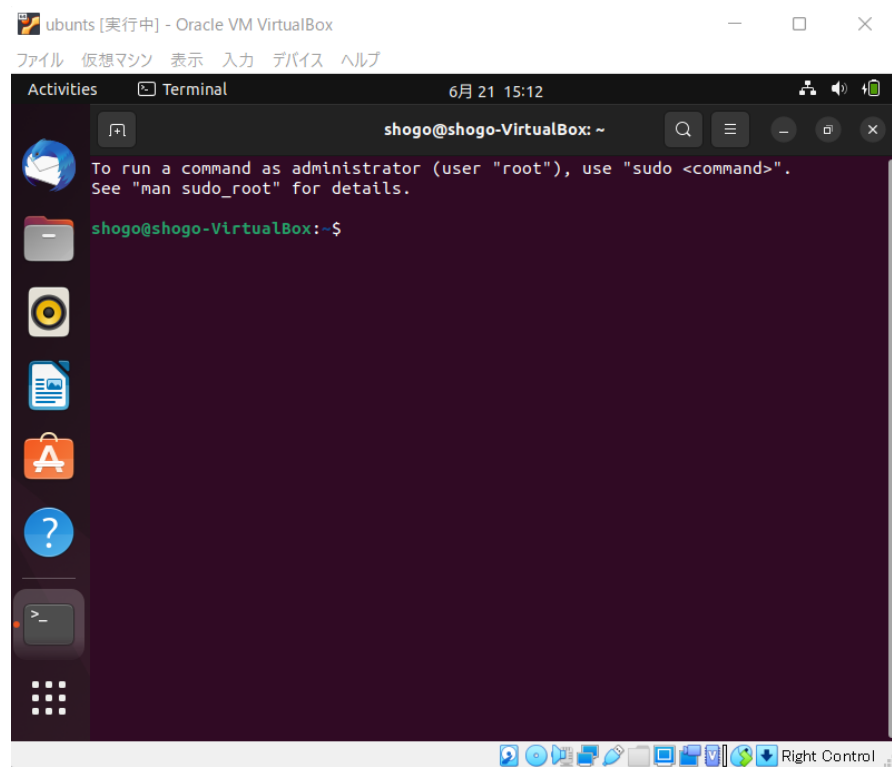
CUIはUnixやLinux、MS-DOSなど文字を使ってコンピュータを操作するタイプのOS

OSの仕事

GUI(Grafical user interface)



CUI(character user interface)



OSの仕事

```
shogo@shogo-VirtualBox: ~  
shogo@shogo-VirtualBox:~$ cat /etc/debian  
cat: /etc/debian: No such file or directory  
shogo@shogo-VirtualBox:~$ cat /etc/debian_version  
bookworm/sid  
shogo@shogo-VirtualBox:~$ cat /etc/lsb-release  
DISTRIB_ID=Ubuntu  
DISTRIB_RELEASE=22.04  
DISTRIB_CODENAME=jammy  
DISTRIB_DESCRIPTION="Ubuntu 22.04 LTS"  
shogo@shogo-VirtualBox:~$ cat /etc/os-release  
PRETTY_NAME="Ubuntu 22.04 LTS"  
NAME="Ubuntu"  
VERSION_ID="22.04"  
VERSION="22.04 (Jammy Jellyfish)"  
VERSION_CODENAME=jammy  
ID=ubuntu  
ID_LIKE=debian  
HOME_URL="https://www.ubuntu.com/"  
SUPPORT_URL="https://help.ubuntu.com/"  
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"  
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"  
UBUNTU_CODENAME=jammy  
shogo@shogo-VirtualBox:~$ cat /proc/version  
bash: cat /proc/version: No such file or directory  
shogo@shogo-VirtualBox:~$ uname -a  
Linux shogo-VirtualBox 5.15.0-39-generic #42-Ubuntu SMP Thu Jun 9 23:42:32 UTC 2023; root:x86_64 GNU/Linux  
shogo@shogo-VirtualBox:~$
```

UbuntuでカーネルやLinuxの詳細を調べました

皆さんのWindowsではwmic os get /format:LISTやVerで出せます

BIOSに関してはwmic bios get sbiosbiosversionで出せます

OSの仕事

API(Application Program Interface)

ソフトウェアの機能や管理するデータなどを、外部の他のプログラムから呼び出して利用するための手順やデータ形式を定めた規約のこと

APIがOSやミドルウェアごとに決まっています、それを呼び出すことで開発が簡単にできる

ボ	フ	デ	フ	フ	ウ	ウ	フ	ス	メ
タ	ア	イ	ア	ア	イ	イ	ア	プ	ッ
ン	イ	レ	イ	イ	ン	ン	イ	レ	セ
描	ル	ク	ル	ル	ド	ド	ル	ッ	ー
画	生	ト	保	読	ウ	ウ	削	ド	ジ
	成	リ	存	込	登	表	除	の	表
		取			録	示		待	示
		得						機	

RPA (Robotic Process Automation)

人に代わってパソコンに自動的に業務をさせる仕組み
入力→検索→集計→登録→送信
普通に行うと単純作業が続く
ヒューマンエラーも起きやすくなるので、PCに手順を自動化して簡単にするソフトウェア

[【2022年】RPAツール比較17選！おすすめサービスの価格や機能の比較表も | BOXIL Magazine](#)

ジョブ管理

ジョブとは...

利用者から見た仕事の単位がジョブです。
ジョブを効率よく処理できるように、OSは実行スケジュールを管理しています。

バッチファイルとジョブ

Windowsのパソコンにもバッチファイルという仕組みがあります。
このバッチファイルをたくさん登録して自動実行させるような仕組みがジョブ管理であるようにとらえればよいです。



突然ですが、それでは
バッチファイルを作っ
てみましょう

手順

・ バッチファイルの作成

- 1.まずテキストエディタを開いてください
- 2.そこに次のスライドのコマンドを入力してみてください
- 3.ファイルから名前を付けて保存を選択してください。
- 4.ファイルの種類を「すべてのファイル」にしてください
- 5.「.bat」の形式で名前を付けてください
- 6.エンコードはANSIにしてください
- 7.保存を押下してください

・ バッチファイルの実行

- 1.保存先のファイルをエクスプローラー画面で出して
- 2.ダブルクリックしてください

ジョブ管理

@echo off ← プログラム全体のエコー機能をオフにしている。

echo バッチファイルの拡張子
は？

echo.

echo 1. bat

echo 2. txt

echo 3. c

echo 4. zip

choice /c 1234 /t 5 /d 2 /n /m "答
え..."

if %errorlevel% equ 1 goto ok

cls

echo 不正解。正解は1番のbatで
す。

pause >nul

exit

:ok

cls

echo 正解。正解は1番のbatです。

pause >nul

exit

@はバックグラウンドで実行

echoは後ろの文字を表示

choice /cで入力キーの指定

/tはタイムリミットを秒数で指定する

/nで入力キーのコードを表示しない

/mで後ろの文字を代わりに表示する

errorlevelは終了コードを取得する

pause は入力があるまでポーズする

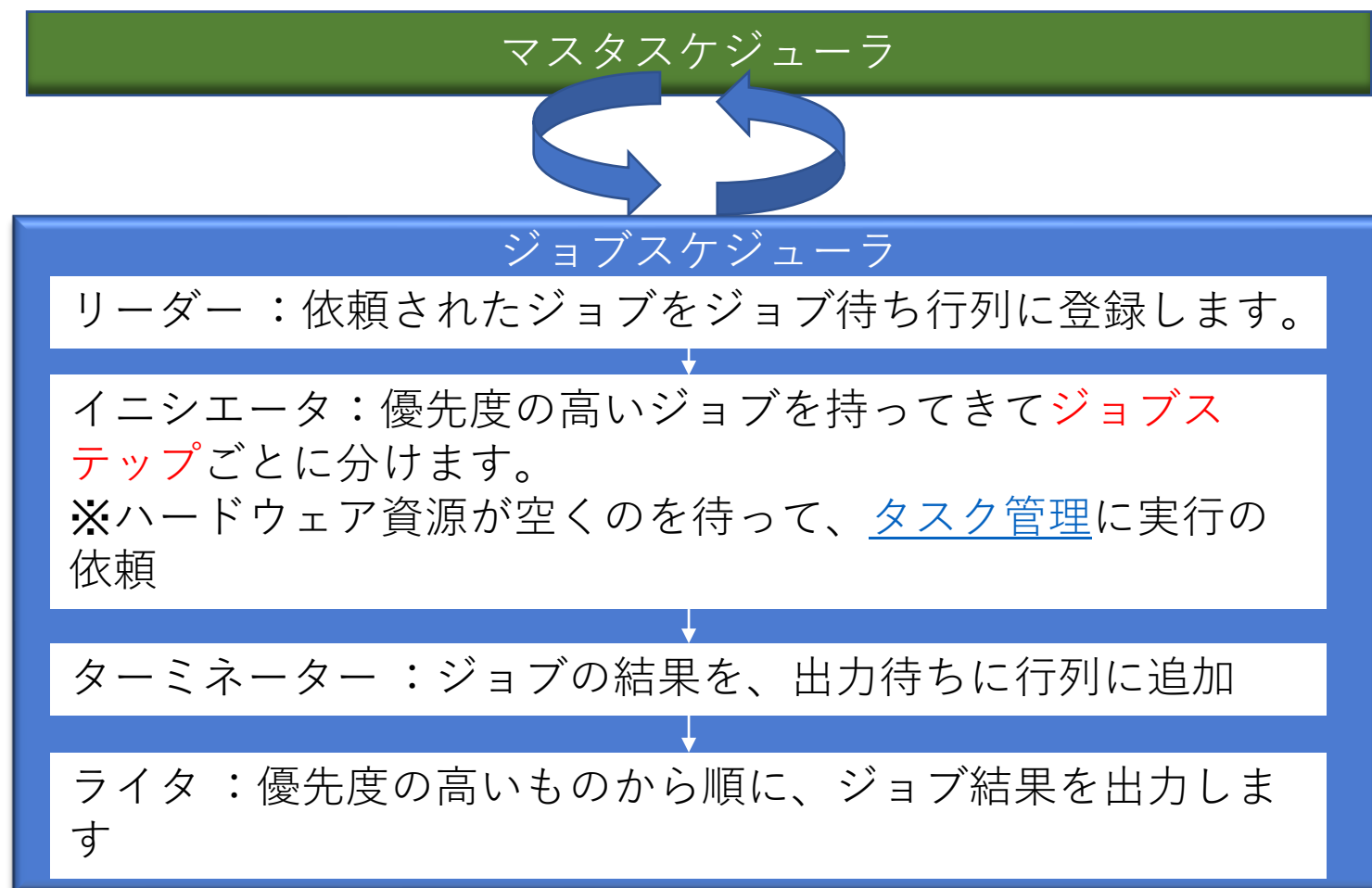
[windowsバッチ コマンド別解説 \(jj-blues.com\)](http://jj-blues.com)

[.bat初心者・未経験者に贈るコマンド集 - Qiita](#)

ジョブ管理

カーネルが持つ機能の一つにマスタスケジューラというプログラムがあります。この、マスタスケジューラに利用者はジョブの実行を依頼します。

ジョブ管理の流れ

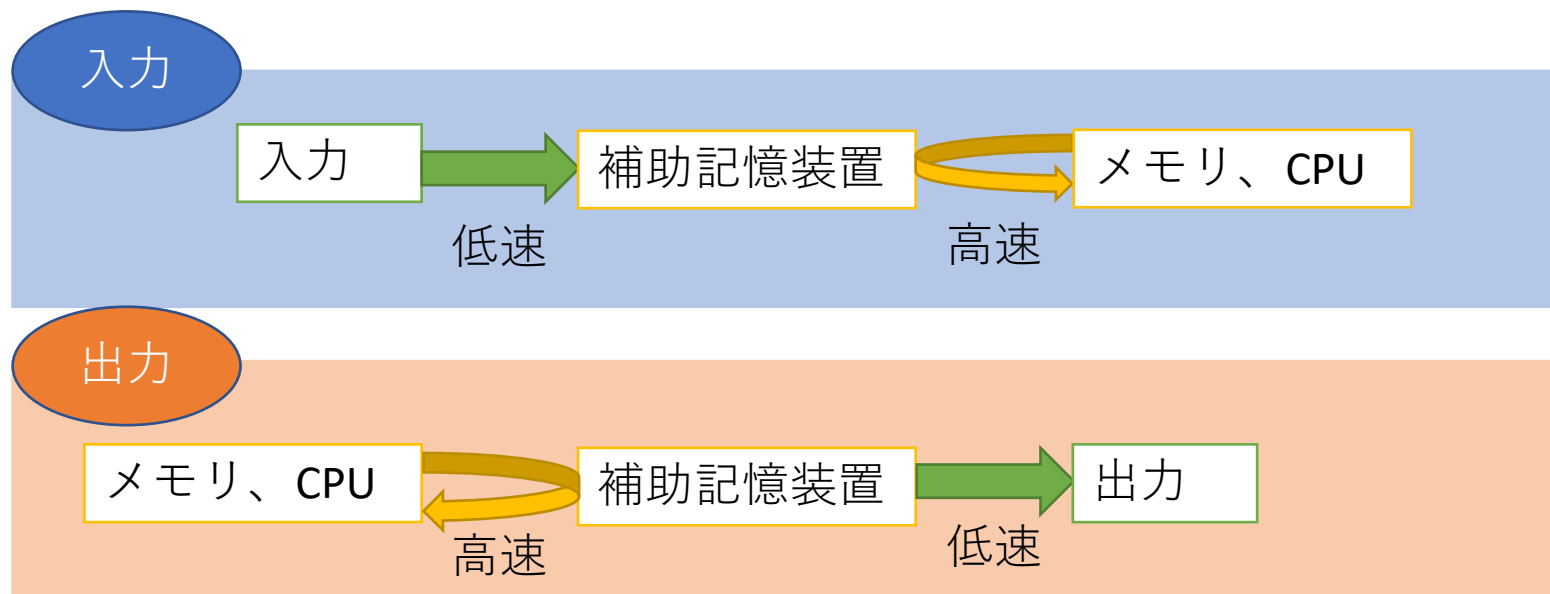


ジョブ管理

スプーリング

ただ、普通に実行すると出力や入力の機会はCPUやメモリ、HDDのように早く動くことができません...

これを解決するために以下のようにすることで効率を高めています。



こうした、「低速な装置とデータのやり取りを高速な補助記憶装置を介して行うことで処理効率を高める方法」をスプーリングと呼びます。

タスク管理

```
shogo@shogo-VirtualBox: ~  
top - 15:46:20 up 36 min, 1 user, load average: 0.00, 0.00, 0.00  
Tasks: 175 total, 1 running, 174 sleeping, 0 stopped, 0 zombie  
%Cpu(s): 2.7 us, 0.7 sy, 0.0 ni, 96.6 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st  
MiB Mem : 971.2 total, 69.6 free, 526.9 used, 374.7 buff/cache  
MiB Swap: 448.4 total, 262.0 free, 186.4 used. 293.9 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1578	shogo	20	0	4052156	229472	75296	S	3.3	23.1	0:27.82	gnome-+
2529	shogo	20	0	571384	51964	39624	S	0.7	5.2	0:01.86	gnome-+
372	systemd+	20	0	14776	3216	2896	S	0.3	0.3	0:02.47	system+
2795	shogo	20	0	22792	4212	3608	R	0.3	0.4	0:00.03	top
1	root	20	0	166588	8112	5416	S	0.0	0.8	0:18.48	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthrea+
3	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_gp
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_pa+
5	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	netns
7	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworke+
10	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_per+
11	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_ta+
12	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_ta+
13	root	20	0	0	0	0	S	0.0	0.0	0:00.18	ksofti+
14	root	20	0	0	0	0	I	0.0	0.0	0:00.81	rcu_sc+
15	root	rt	0	0	0	0	S	0.0	0.0	0:00.02	migrat+
16	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_i+
17	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
18	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kdevtm+
19	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	inet_f+
20	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kauditd
21	root	20	0	0	0	0	S	0.0	0.0	0:00.00	khungt+

タスクとは実行中のプログラムです。CPUは複数のことを同時に処理できるわけではないです。

ただ、タスク管理によって使用権をタスク間で持ち回せたり、割り込みを処理したりすることで実行できているのです。

タスク管理

```
shogo@shogo-VirtualBox: ~  
top - 15:46:20 up 36 min, 1 user, load average: 0.00, 0.00, 0.00  
Tasks: 175 total, 1 running, 174 sleeping, 0 stopped, 0 zombie  
%Cpu(s): 2.7 us, 0.7 sy, 0.0 ni, 96.6 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st  
MiB Mem : 971.2 total, 69.6 free, 526.9 used, 374.7 buff/cache  
MiB Swap: 448.4 total, 262.0 free, 186.4 used. 293.9 avail Mem  
  
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM    TIME+  COMMAND  
 1578 shogo     20   0 4052156 229472 75296 S   3.3   23.1   0:27.82 gnome-+  
 2529 shogo     20   0 571384   51964 39624 S   0.7    5.2   0:01.86 gnome-+  
   372 systemd+  20   0  14776    3216  2896 S   0.3    0.3   0:02.47 system+  
 2795 shogo     20   0 22792    4212  3608 R   0.3    0.4   0:00.03 top  
    1 root       20   0 166588    8112  5416 S   0.0    0.8   0:18.48 systemd  
    2 root       20   0      0      0      0 S   0.0    0.0   0:00.00 kthrea+  
    3 root        0 -20      0      0      0 I   0.0    0.0   0:00.00 rcu_gp  
    4 root        0 -20      0      0      0 I   0.0    0.0   0:00.00 rcu_pa+  
    5 root        0 -20      0      0      0 I   0.0    0.0   0:00.00 netns  
    7 root        0 -20      0      0      0 I   0.0    0.0   0:00.00 kworke+  
   10 root        0 -20      0      0      0 I   0.0    0.0   0:00.00 mm_per+  
   11 root       20   0      0      0      0 S   0.0    0.0   0:00.00 rcu_ta+  
   12 root       20   0      0      0      0 S   0.0    0.0   0:00.00 rcu_ta+  
   13 root       20   0      0      0      0 S   0.0    0.0   0:00.18 ksofti+  
   14 root       20   0      0      0      0 I   0.0    0.0   0:00.81 rcu_sc+  
   15 root       rt   0      0      0      0 S   0.0    0.0   0:00.02 migrat+  
   16 root      -51   0      0      0      0 S   0.0    0.0   0:00.00 idle_i+  
   17 root       20   0      0      0      0 S   0.0    0.0   0:00.00 cpuhp/0  
   18 root       20   0      0      0      0 S   0.0    0.0   0:00.00 kdevtm+  
   19 root        0 -20      0      0      0 I   0.0    0.0   0:00.00 inet_f+  
   20 root       20   0      0      0      0 S   0.0    0.0   0:00.00 kauditd  
   21 root       20   0      0      0      0 S   0.0    0.0   0:00.00 khungt+
```

項目	概要
175 total	合計タスク数
1 running	稼働中タスク
174 sleeping	待機中タスク
0 stopped	停止タスク
0 zombie	ゾンビタスク

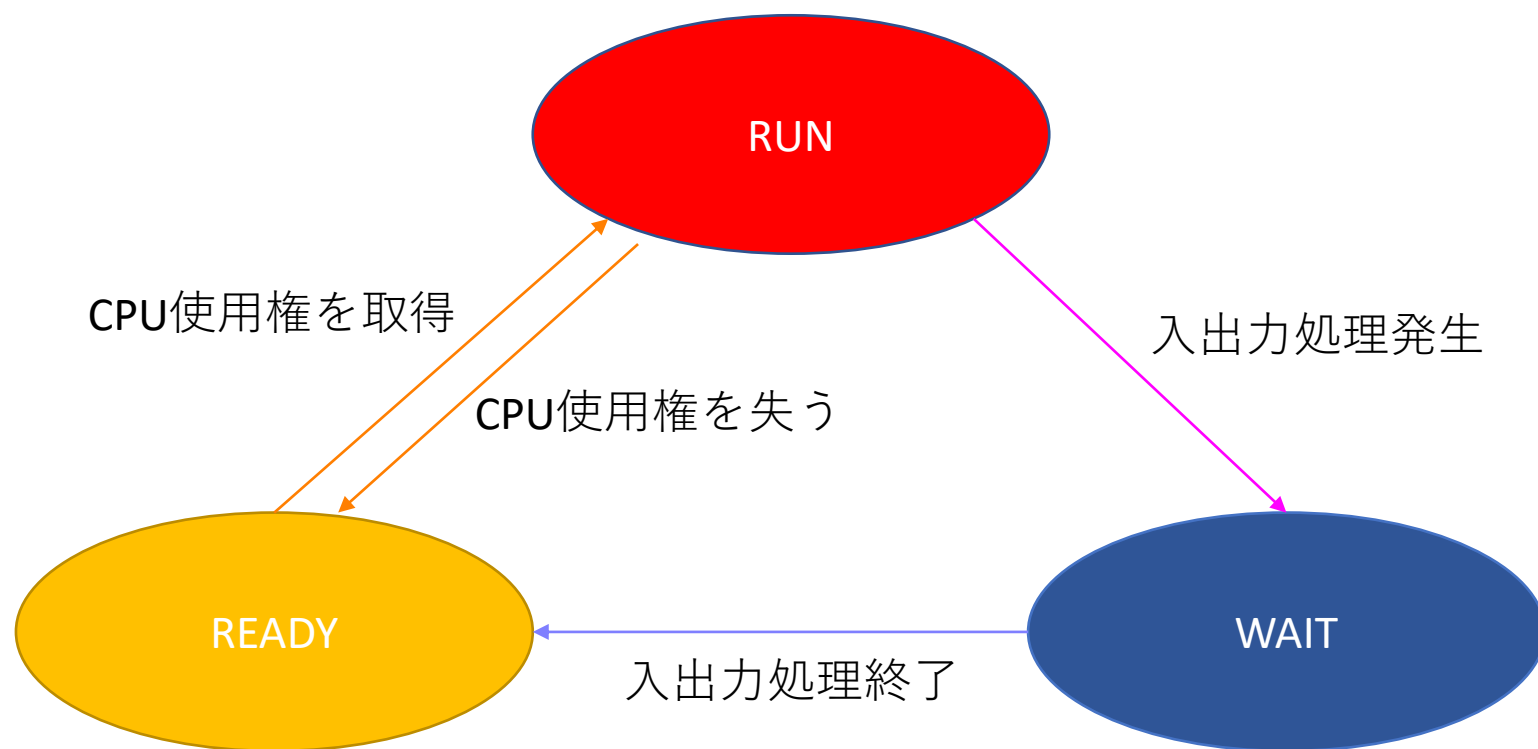
教科書に合わせて説明すると

- run=running
- ready
- wait ==sleeping

教科書に書いていないものだと

- stopped 停止しているタスク
- zombie 終了しているのに動き続けているタスク

タスク管理



WAITからRUNに直接は移動できないのでREADYを挟んで実行している。
RUN中に入力出力があった場合にWAITになる(cf.スプーリング)
入出力が終わったらWAIT→READYになる。

タスク管理

ディスパッチャとタスクスケジューリング

ディスパッチャとはCPUの使用権を割り当てるのかを管理しているプログラムのことです。

どのタスクを処理するかを決めるのに実行順序を決める必要が出てきます。これを**タスクスケジューリング**と呼びます

タスクスケジューリングについて、いくつか見てみましょう

到着順方式

実行可能になったタスク順に、CPUの使用権を割り当てる方式です。タスクに優先度がないため、実行の途中でCPU使用権が奪われることが無い(ノンプリエンション：OSの介入なし)

コンテキスト切り替え

ノンプリエンティブ方式

優先順(プライオリティ順)方式

タスクにそれぞれ優先度を設定し、優先度が高いものから順に処理をする。

CPUの使用権の優先度が高い命令が入ると低い命令はCPUの使用権を奪われます(プリエンション：OSの介入あり)

プリエンティブ方式

ラウンドロビン方式

CPUの使用権を一定時間に変える方式

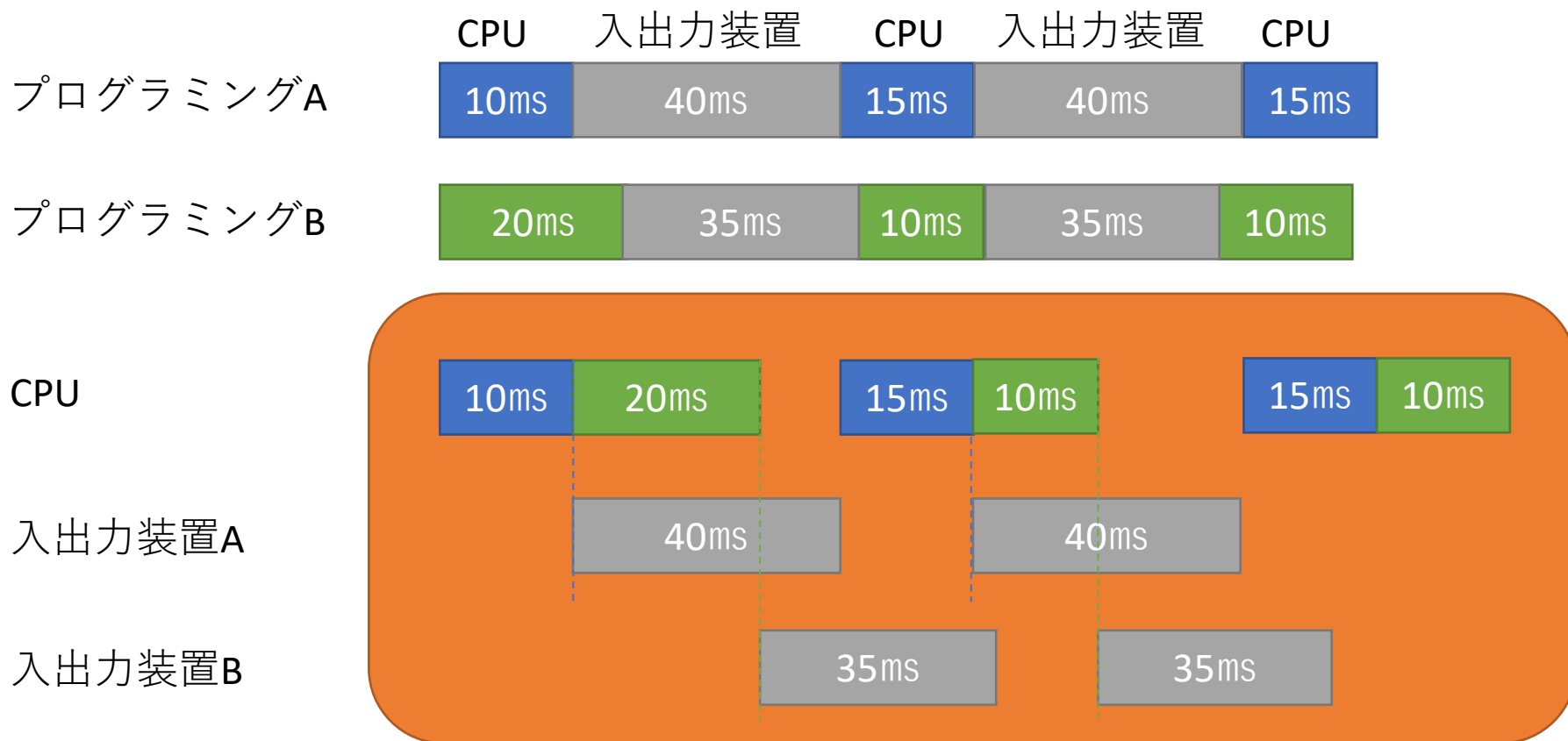
プリエンティブ方式

タスク管理

マルチプログラミング

タスク管理の役割はCPUの有効活用につきます。そのため、できるだけ遊休時間を最小にする必要があります...

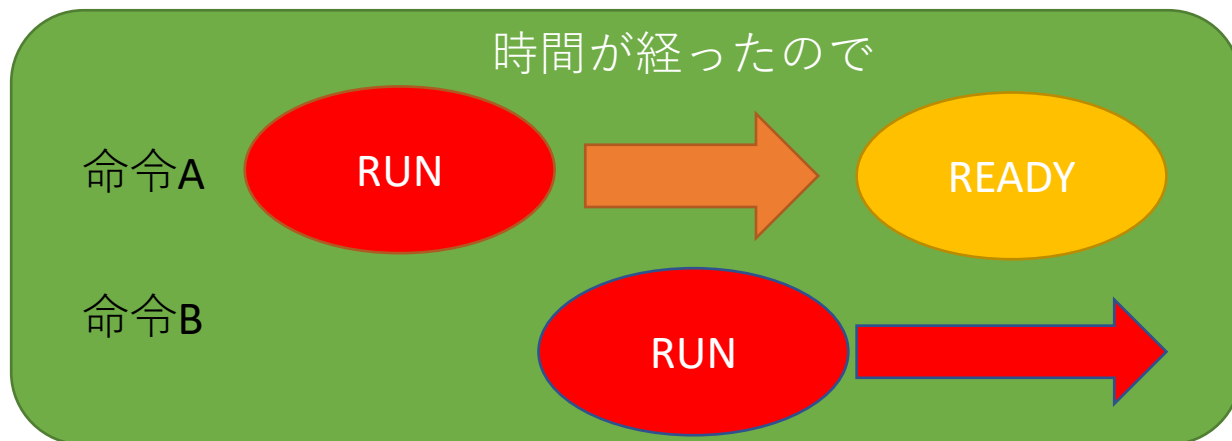
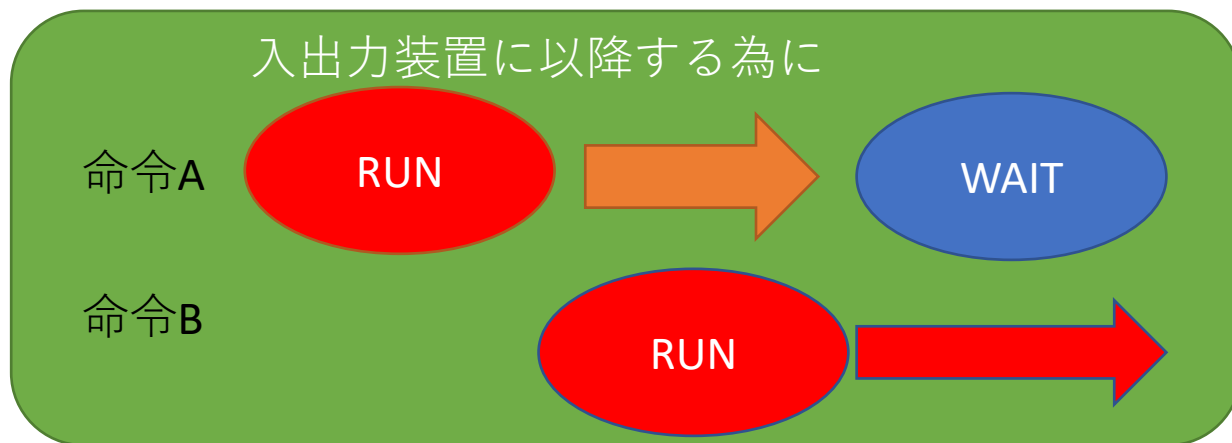
では、複数の命令があった場合は...



タスク管理

割り込み処理

実行中のタスクを中断して、別の処理に切り替え、そのタスクが終わると元のタスクに戻ることを



など

タスク管理

内部割込み ・ ・ ・ プログラムの異常などによって起きる割込み		
	プログラム割込み	ゼロによる除算や桁あふれ(オーバーフロー)、仮想記憶において存在しないページへのアクセスなど予期しないことが起きた時の割込み
	SCV(Super Visor Call)割込み	入出力処理を要求するなど、カーネル呼び出し命令が発行されたときに生じる割込み
外部割込み ・ ・ ・ 機械などの外部的な要因によって起きる割込み		
	入出力割込み	入出力装置の動作完了時や中断時生じる割込み
	機械チェック割込み	電源の異常や主記憶装置の障害など、ハードウェアの異常発見時に生じる割込み。
	機械チェック割込み	電源の異常や主記憶装置の障害など、ハードウェアの異常発見時に生じる割込み。
	コンソール割込み	オペレータ(利用者)による介入が行われたときに生じる割込み。
	タイマ割込み	規定の時間を過ぎたときに生じる割込み

```
shogo@shogo-VirtualBox:~$ dir
```

```
Desktop Downloads Pictures Templates snap
```

```
Documents Music Public Videos
```

```
shogo@shogo-VirtualBox:~$ ls -al
```

```
total 84
```

```
drwxr-x--- 14 shogo shogo 4096 6月 24 15:20 .
```

```
drwxr-xr-x  3 root  root  4096 6月 21 14:49 ..
```

```
-rw-----  1 shogo  shogo    512 6月 23 14:02 .bash_history
```

```
-rw-r--r--  1 shogo  shogo    4096 6月 21 14:49 .bash_logout
```

```
-rw-r--r--  1
```

```
-rwx----- 1
```

```
-rwx----- 1
```

```
-rw-----
```

```
-rwx----- 3 shogo shogo 4096 6月 21 15:10 .local
```

```
-rw-r--r--  1 shogo shogo  354 6月 24 15:20 .pam_environment
```

```
-rw-r--r--  1 shogo shogo  807 6月 21 14:49 .profile
```

```
-rw-rw-r--  1 shogo shogo   66 6月 23 11:15 .selected_editor
```

```
-rw-r--r--  1 shogo shogo    0 6月 23 11:16 .sudo_as_admin_successful
```

```
drwxr-xr-x  2 shogo shogo 4096 6月 21 15:10 Desktop
```

```
drwxr-xr-x  2 shogo shogo 4096 6月 21 15:10 Documents
```

付録

ファイル管理

ファイルの種類

代表的なファイル形式

- ・テキスト形式(.txt)
- ・CSV形式 ・ ・ ・ テキスト形式ですが個々のデータをカンマで区切り表形式のファイルとしたもの

・PDF

画像用

- ・BMP
- ・JPEG
- ・GIF
- ・PNG

音声

- ・MP3
- ・MIDI
- ・WAV

動画

MPEG

圧縮 ・ ・ ・ 例として1がいくつ入っているか0がいくつ入っているかを表して圧縮する方法がある

- ・Zip
- ・7zなど

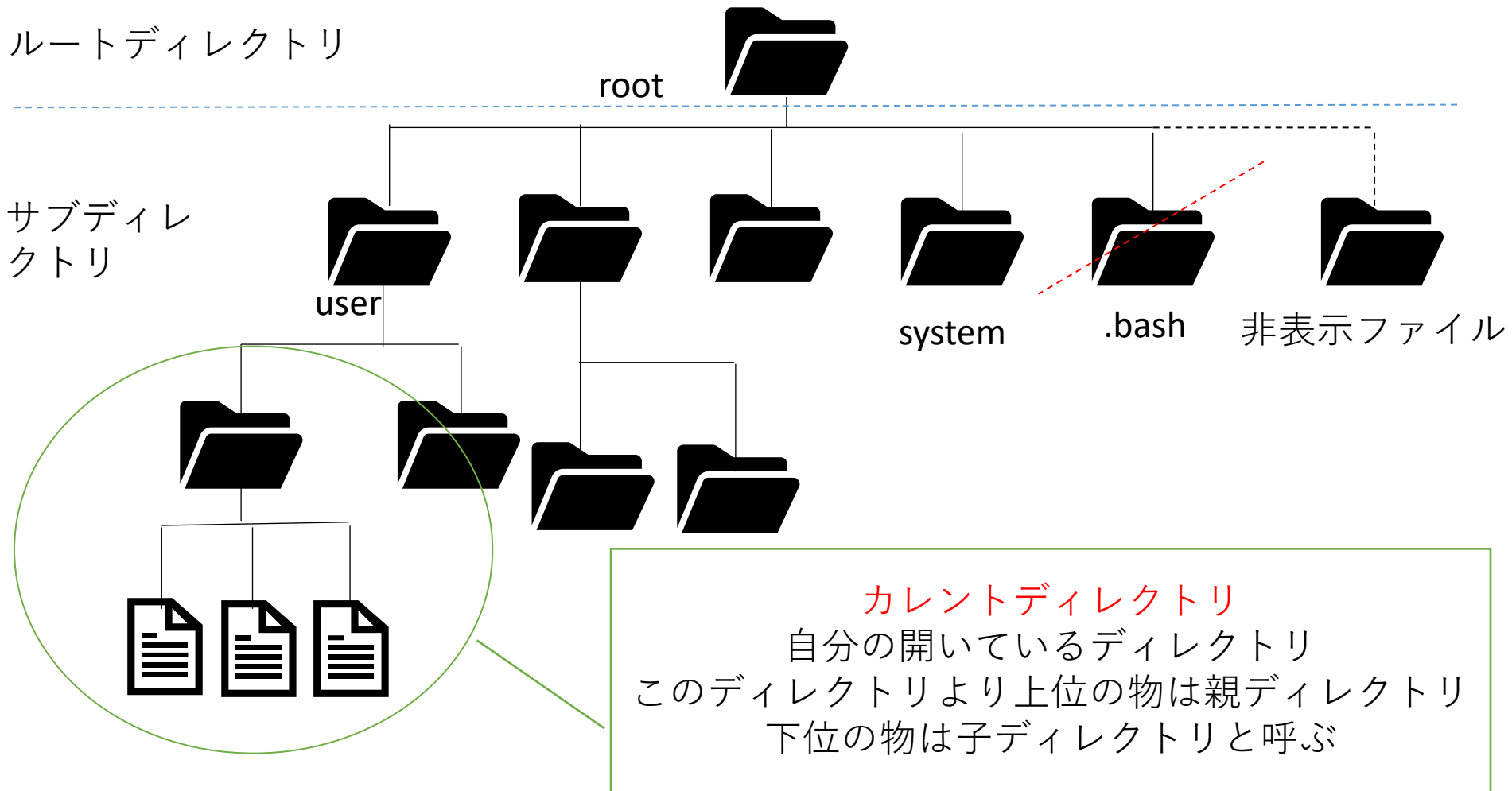
元に戻そうとそしても元に完全に戻らない(不可逆圧縮)戻る場合(可逆圧縮)

ディレクトリ構造

ファイルはデータが入っている場所であるがこれをまとめたものをディレクトリと呼び階層構造を取っている。

また、ファイルの場所を記した物をパスと呼ぶ

Cf. windows: ¥user¥name¥temp 、 Linux: /user/name/temp



レコードとアクセス方法

実はファイルはレコードと呼ばれる物の集合体です。
このレコードは1つのデータだと思っていただければと思います。
レコードの形式はアプリケーションソフトによってまちまちです。
また、このレコードへのアクセス方法について記述します。

順次アクセス



直接アクセス



動的アクセス



レコードとアクセス方法(編成)

順編成ファイル



順次アクセスのみ可能

頭から順番にレコードを記録していく順編成ファイル

もっとも単純な編成法で、順次アクセスのみが可能

直接編成ファイル

直接アドレス方式



直接アドレス方式はキーの内容をそのまま取り出す

レコードとアクセス方法(編成)

直接編成ファイル

間接アドレス方式

ハッシュ関数



会社	部署	社員番号	名前
A社	代表	A000001	田中一郎
A社	総務部	A000033	山本二郎
A社	開発部	A000128	佐藤三郎
A社	開発部	A000130	鈴木四郎

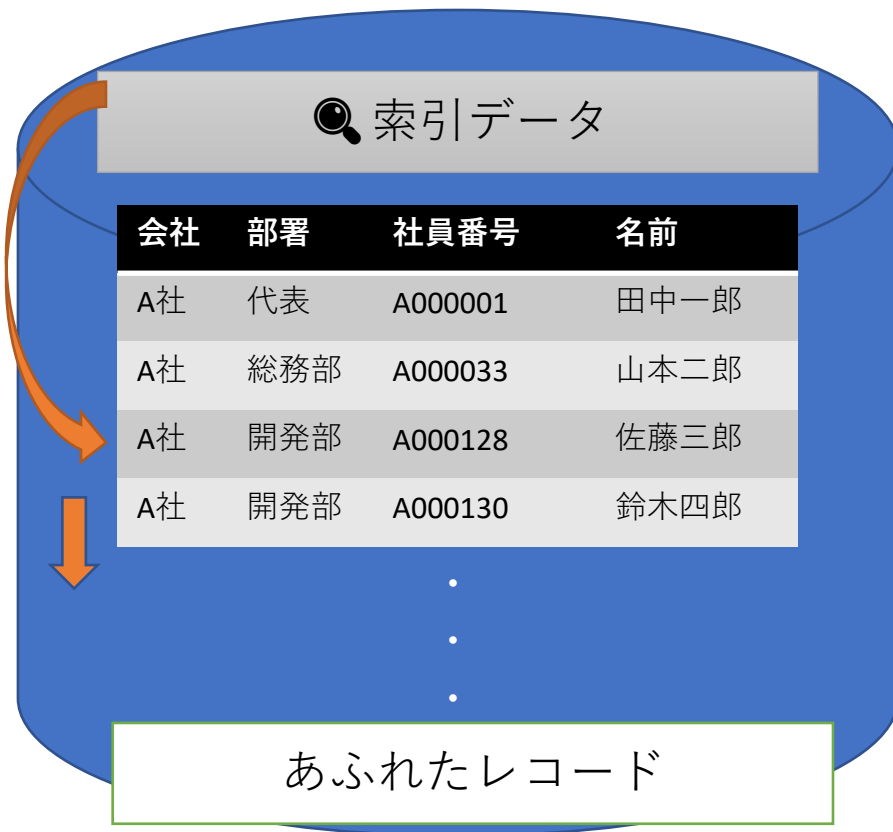
ハッシュ関数という計算式により、キー値から格納アドレスを算出して用いる

ハッシュ関数...

入力値の長さによらずあらかじめ決められた固定長の出力データを得る関数

レコードとアクセス方法(編成)

索引編成ファイル



索引域

基本データ域

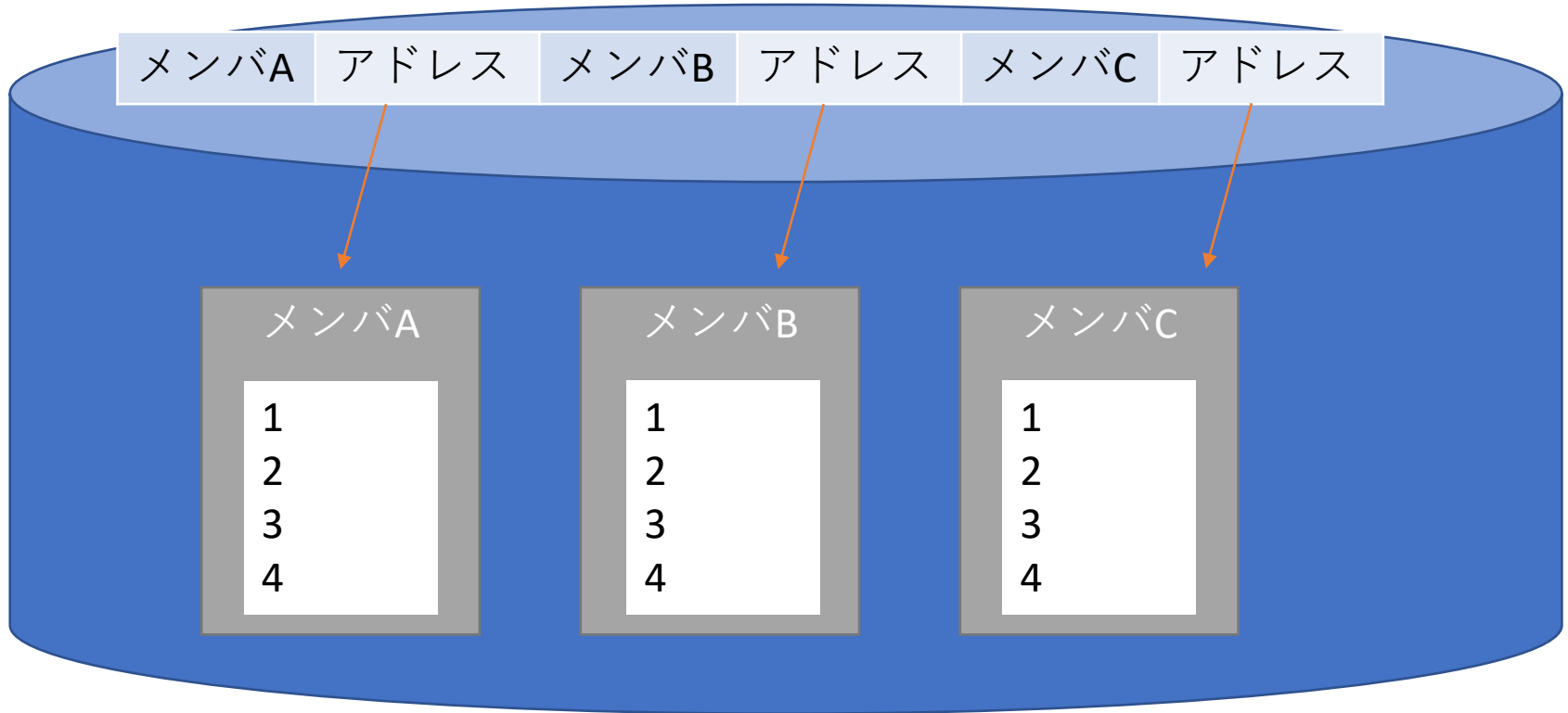
あふれ域

索引域と基本データ域、あふれ域という3つの領域から構成され

索引による直接アクセスと先頭から順次アクセスという両方の特性を備える編成法です

レコードとアクセス方法(編成)

区分編成ファイル



メンバと呼ばれる順編成ファイルを複数持ち、それらをかくのうするめんばいきと、各メンバへのアドレスを管理するディレクトリ域で構成される編成法