

# オペレーティングシステム デーモン・サービス

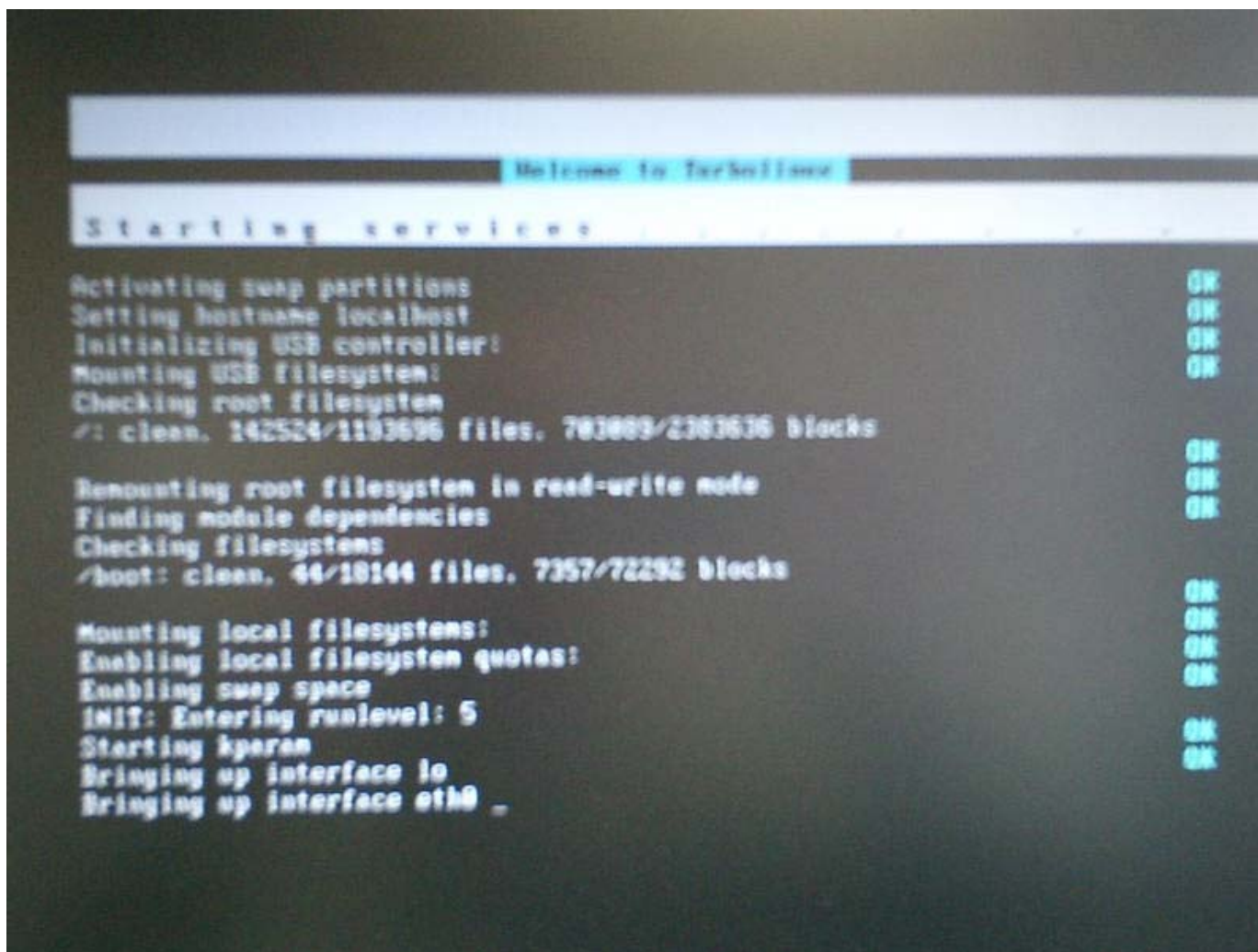
2005年1月21日

海谷 治彦

# 目次

- OS(Kernel)起動後の挙動 ～RedHat系の場合
  - /etc/inittab
  - /etc/rc.d/ ランレベルの話
- 主な初期化プロセスの解説
  - network
  - portmap
  - syslog
  - .....

# Linuxを起動すると・・・

A screenshot of a Linux terminal window showing the boot process. The terminal has a dark background with light blue and white text. At the top, there's a light blue banner with the text "Welcome to Terbolinux". Below that, another light blue banner says "Starting services". The main text shows various system initialization steps, each followed by "OK" in light blue on the right side. The steps include activating swap partitions, setting hostname to localhost, initializing USB controller, mounting USB filesystem, checking root filesystem (reporting clean), remounting root filesystem in read-write mode, finding module dependencies, checking filesystems (reporting clean for /boot), mounting local filesystems, enabling local filesystem quotas, enabling swap space, entering runlevel 5, starting kperon, and bringing up network interfaces lo and eth0.

```
Welcome to Terbolinux

Starting services

Activating swap partitions                                OK
Setting hostname localhost                                OK
Initializing USB controller:                              OK
Mounting USB filesystem:                                  OK
Checking root filesystem
/: clean. 142524/1193696 files. 783889/2383636 blocks
                                                         OK
Remounting root filesystem in read-write mode              OK
Finding module dependencies                               OK
Checking filesystems
/boot: clean. 44/18144 files. 7357/72292 blocks
                                                         OK
Mounting local filesystems:                               OK
Enabling local filesystem quotas:                         OK
Enabling swap space                                       OK
INIT: Entering runlevel: 5                                OK
Starting kperon                                           OK
Bringing up interface lo                                  OK
Bringing up interface eth0 _
```

# Linuxを停止すると・・・

```
Linux 2.4.18-1 on an i386 (localhost)
VG: vc/1

INIT: Switching to runlevel: 6
INIT: Sending processes the TERM signal
Shutting down cupsd:
Stopping keytable
Shutting down lilm:
Stopping canna
Stopping alsasound
Shutting down sshd:
Stopping INET services:
Stopping at daemon:
Stopping crond:
Stopping hotplug:
Saving random seed
Shutting down interface eth0
Stopping pcnic
Stopping kparan
Shutting down kernel logger:
Shutting down system logger:
Starting killall
Sending all processes the TERM signal...
INIT: no more processes left in this runlevel
```

[illegible]

# 本日の観点は？

- この「だらだら」表示されるのは何をやっているかを知る.
- さらに, Kernel以外に必要な常時動作している処理について学ぶ.

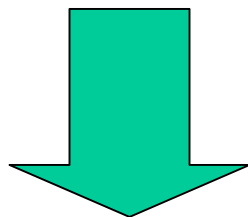
復習

# カーネルの機能

- プロセス・リソース管理
- メモリ管理
- デバイス管理
- ファイル管理

# カーネルだけでは役不足？

- ネットワークにつながらない.
- プリンタが機能しない.
- そもそも, ログインできない.....



- いわゆる環境設定というのがほとんど何もおこられない.

# initからの追加処理起動

- Linux(Unix)では最初のプロセスinitから、カーネルの仕事を助けるプロセスを自動的に起動することができる.
  - このようなプロセスを通常 **デーモン(daemon)** とLinuxでは呼ぶ.
- どのプロセスをどんな順序で呼ぶかは、テキストファイルに平易に設定されている.
  - ⇒ コンパイルのし直し等が不要



# 最初のプロセス

- 複製をもとにプロセスが生成されると、最初にタネになるプロセスがないとはじまらない.
- Linuxには以下の2つのタネになるプロセスがある.
  - プロセス0 Swapper, 初期化プロセス等とよばれ, カーネル内の変数等の初期化をする.
  - プロセス1 Init ほとんどすべてのプロセスの先祖となる

# 最初のプロセスの実際

- プロセス0
  - main.c の 1355行目が処理実体
  - sched.c の 97行目で配列の1個目要素としている.
- プロセス1
  - main.c の1441行から呼び出される.
  - 実体は, 1601行目
  - 1474行のdo\_basic\_setup を介して, kflushd, kupdate, kpiod, kswapd等, 基盤となるプロセスを開始しているのが読める, 1536行目あたり.

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	1	0	0	Aug27	?	00:00:05	init
root	2	1	0	Aug27	?	00:00:00	[kflushd]
root	3	1	0	Aug27	?	00:00:01	[kupdate]
root	4	1	0	Aug27	?	00:00:00	[kpiod]
root	5	1	0	Aug27	?	00:00:04	[kswapd]

# /etc/inittab

- プロセス init から呼び出される処理の初期設定が記載されている.

```
# inittab      This file describes how the INIT process should set up
#              the system in a certain run-level.
```

中略

```
id:3:initdefault:
```

```
# System initialization.
```

```
si::sysinit:/etc/rc.d/rc.sysinit
```

```
10:0:wait:/etc/rc.d/rc 0
```

以後略

# runlevel

- OSの用途や状況によって, initから起動される処理を6通り事前に準備している.
- デフォルト値を inittab中に設定できる.

```
# Default runlevel. The runlevels used by RHS are:  
# 0 - halt (Do NOT set initdefault to this)  
# 1 - Single user mode  
# 2 - Multiuser, without NFS (The same as 3, if you do not have networking)  
# 3 - Full multiuser mode  
# 4 - unused  
# 5 - X11  
# 6 - reboot (Do NOT set initdefault to this)  
#  
id:3:initdefault:
```

サーバー等は3, クライアントは5にする場合が多い.

# runlevel 3

- Full multiuser mode
- OSを複数のユーザーが利用可能な完全な状態で起動する.
- ただし, Window System は自動に起動しない.
  - LinuxはWindowがなくても作業できるし.
- コンソールから直接に操作をしないマシン(サーバー等)の場合, このレベルをデフォにしておく.
- 私は常にこのレベルにしている.

# runlevel 5

- Full multiuser mode with X11
- level 3に加えWindow System(X11)を自動起動する.
- 昨今のPC-Linuxはこの設定が主流.
  - Windows的に使う人が多いため.
- Window Sys.の起動が遅いので、一般に再起動に時間がかかってウザい.

## /etc/rc.d/rc?.d/

- 各runlevelで実行される実行ファイルの置き場.
- 実行ファイルはshell scriptへのシンボリックリンク
- ファイル名は以下のどちらか
  - S番号名前
    - 例 S60lpd
  - K番号名前
    - 例 K51sshd
- 基本的にUNIXの中でもSVR4(Solaris)に構造が  
にているので私としては好き. BSD Unixとはちよつ  
と違う.

# S/Kファイルの実行順序

- 設定されたrunlevel内のファイルが実行される.
  - levelが5なら, /etc/rc.d/rc5.d/ の下
- OS起動時に, Sから始まる番号の若いものから順に実行される.
- OS終了時には, Kからはじまる番号の若いものから実行される.
- ……という風に, /etc/rc ファイルに記述される.
- S=Start, K=Kill の略



# 各スクリプトの一般構造

- shell script である.
  - 知ってるよね？
- 引数として, start と stop によって動作が変わるように最低でも記述されている.
- S の場合は start 側が, Kの場合はstop側が実行される.

# 例 (httpd)

```
# Source function library.  
./etc/rc.d/init.d/functions
```

```
# See how we were called.
```

```
case "$1" in
```

```
start)
```

```
    echo -n "Starting httpd: "  
    daemon httpd  
    touch /var/lock/subsys/httpd  
    ;;
```

```
stop)
```

```
    echo -n "Shutting down http: "  
    killproc httpd  
    rm -f /var/lock/subsys/httpd  
    rm -f /var/run/httpd.pid  
    ;;
```

```
中略
```

```
*)
```

```
    echo "Usage: $0 {start|stop|restart|reload|status}"  
    exit 1
```

```
esac
```

```
exit 0
```

# 以降 重要なデーモンの個別紹介

# network

- ネットワークインタフェース(ethernet等)を起動するデーモン.
- こいつが動作しないと通信プログラムは一切動かない.
- 具体的な処理内容は,
  - ホスト名, データの送り先(gateway)の情報を取得.
  - マシンに接続されているインタフェースの情報を確認(名前, IPアドレス, マスク等)
  - ifconfigコマンドで取得した情報に従いインタフェースを動作可能状態にする.
- このデーモンが実行されてはじめて, TCP/IP通信が可能となる.

# portmap

- 他のマシンからの手続き呼び出し(RPC)の設定補助デーモン.
  - RPCはマシンの中で番号付けされて管理されており, これをポート番号と言う.
    - 有名なサービスには共通の番号を使うように推奨されている.
      - wwwは80番, メール送信が25番等
  - portmapは, あるRPCにポートと対応付ける(mapする)ことで, 通信ができるように補助してくれるサービス.

# inet

- 他のマシンからの要求に応じて、任意のサービスをオンデマンドで起動するためのデーモン。
  - 動作中は全ての通信内容を観察し、
  - あるポートに新しい接続要求が入ると、
  - その要求に応じたデーモンを起動して、
  - そのポートの処理要求を起動したデーモンに委譲する。
  - /etc/inetd.conf に設定がある。
- あまりセキュアでないので、最近は使われない。

# POPの例

SMTPサーバー(sendmail)と異なり, 常時, 動いているわけではなく, inetd を介して必要な時に起動される.

```
##
## pop
##
pop-2    stream  tcp  nowait  root    /opt/etc/ipop2d    ipop2d
#pop3    stream  tcp  nowait  root    /opt/etc/popper    popper -s
pop3     stream  tcp  nowait  root    /opt/etc/ipop3d    ipop3d
##
## IMAP4
##
imap     stream  tcp      nowait  root    /opt/etc/imapd     imapd
##
## TME 10 Framework daemon
/etc/inetd.conf line 149/183 byte 6922/7280 95% code ASCII (press R
```

# POPでメールを読んでいる状態

```
##
## pop
##
pop-2    stream  tcp  nowait  root    /opt/etc/ipop2d    ipop2d
#pop3    stream  tcp  nowait  root    /opt/etc/popper    popper -s
pop3     stream  tcp  nowait  root    /opt/etc/ipop3d    ipop3d
##
## IMAP4
##
imap     stream  tcp      nowait  root    /opt/etc/imapd     imapd
##
## TME 10 Framework daemon
##
/etc/inetd.conf line 149/183 byte 6922/7280 95% code ASCII (press R)
```

squid	562	512	0	11月	21	?	0:00	/www/squid/bin/ftpget -
squid	560	512	0	11月	21	?	0:00	(dnsserver)
root	566	307	0	11月	21	?	0:00	na_event
root	20973	466	0	12月	01	?	0:01	/usr/local/sbin/sshd2
kaiya	15871	307	0	19:08:50	?	?	0:00	<b>ipop3d</b>
nobody	639	619	0	11月	21	?	0:00	TME_sched
root	619	1	0	11月	21	?	0:00	oserv -p 94 -k /apl/Tiv



# リモートログインの例

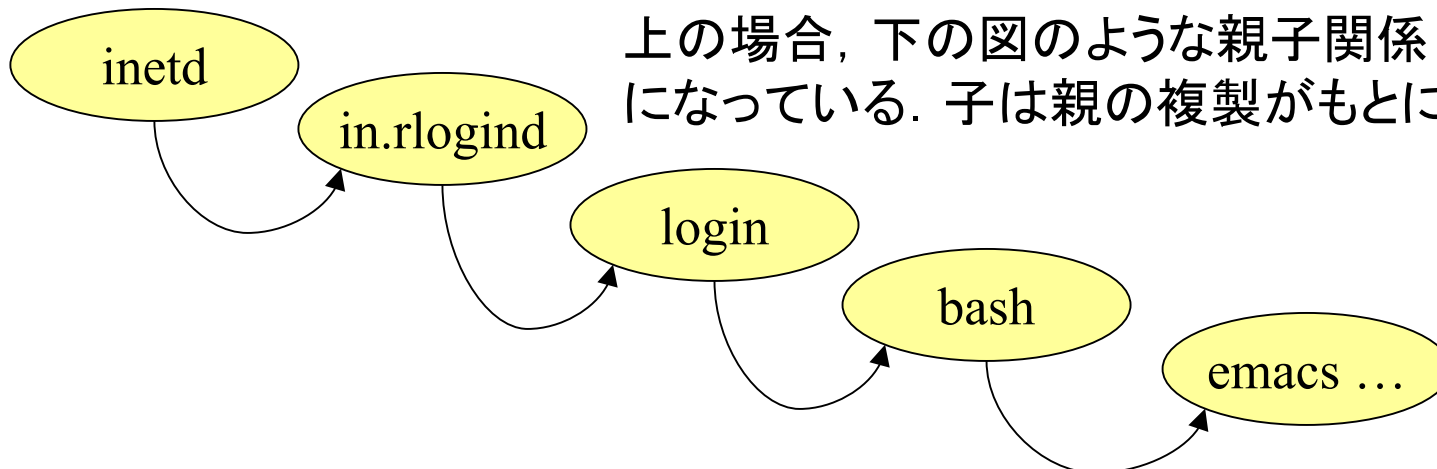
UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	538	1	0	Aug27	?	00:00:00	inetd
root	983	538	0	23:22	?	00:00:00	in.rlogind
root	984	983	0	23:22	pts/1	00:00:00	login -- kaiya
kaiya	985	984	0	23:22	pts/1	00:00:00	-bash
kaiya	1012	985	3	23:23	pts/1	00:00:00	emacs Foo.java

1行が1プロセス

自プロセスの番号

親プロセスの番号

プロセスのもととなったコマンド名



# syslog

- 他のデーモン等が作成した実行記録(ログ)を記録・管理するデーモン.
- 不正アクセス等の兆しや証拠はこのログから分析することもある.
- syslogの設定にもよるが, 通常, /var/log/ディレクトリ下にログを保存する.

# lpd

- 印刷要求の処理をするデーモン.
- 複数の印刷依頼が異なるユーザーから同時に押し寄せても, ちゃんと待ち行列にならべて, ばらばらにならないようにする.
- 他のマシンにつながっているプリンタへの印刷依頼の窓口も行う.

# xntpd

- 他のマシンと時計合わせをするためのデーモン.
- 異なるマシン間で時計が狂ってるとなにかと不便なので, 是非, つかいたい.
- 本学科内では, dns.cs. が正確な時間(GPSから取得)を発信しているのでこいつに合わせるのが良い.

# cron

- 定期的にコマンドを実行するためのシステム.
- ユーザー別に設定される.
- UNIX系には大抵ある.
- Winでいう所の「タスク」に相当.
- 詳しくは man cron を参照

# 実行される内容の確認・記述

```
dotcompts5 /home/kaiya
dotcom% crontab -l
0 4 * * * $HOME/cron/report-backup.sh
0 6 * * * $HOME/cron/selfback-day.sh
30 6 7 * * $HOME/cron/selfback-month.sh
0 9 * * * $HOME/cron/asahi-news.sh
dotcom% █
```

こっちの場合4つのコマンドを、

- 毎日午前4時
- 毎日午前6時
- 毎月7日午前6時半
- 毎日9時

に自動的に実行する。

```
edws2pts2 /home/staff/kaiya
edws2% crontab -l
crontab: can't open your crontab file.
edws2% █
```

設定してないと、  
その旨が表示される。

# アプリ寄りのデーモン

- アプリケーションの動作を助けるもの.
- ま, どこまでがアプリかはクリアでないが.

# sshd

- 暗号化された安全なリモートログインをサービスするデーモン.
- 昔のrshdやrlogindの代わり.

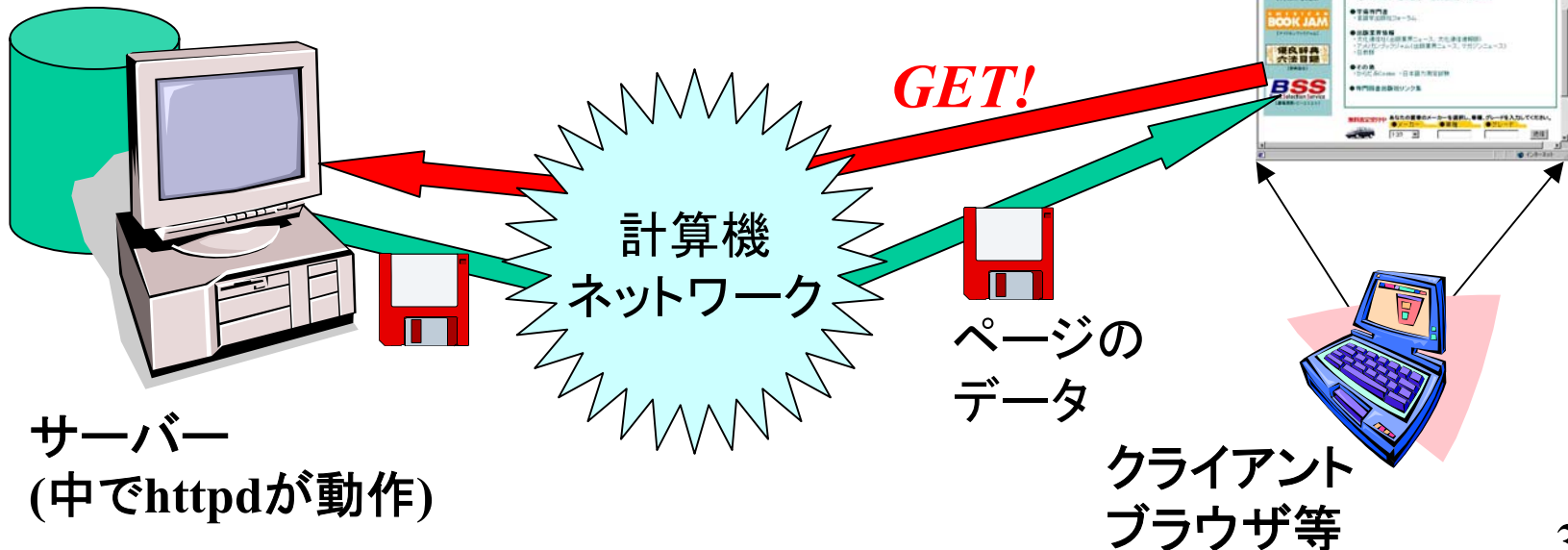


# httpd

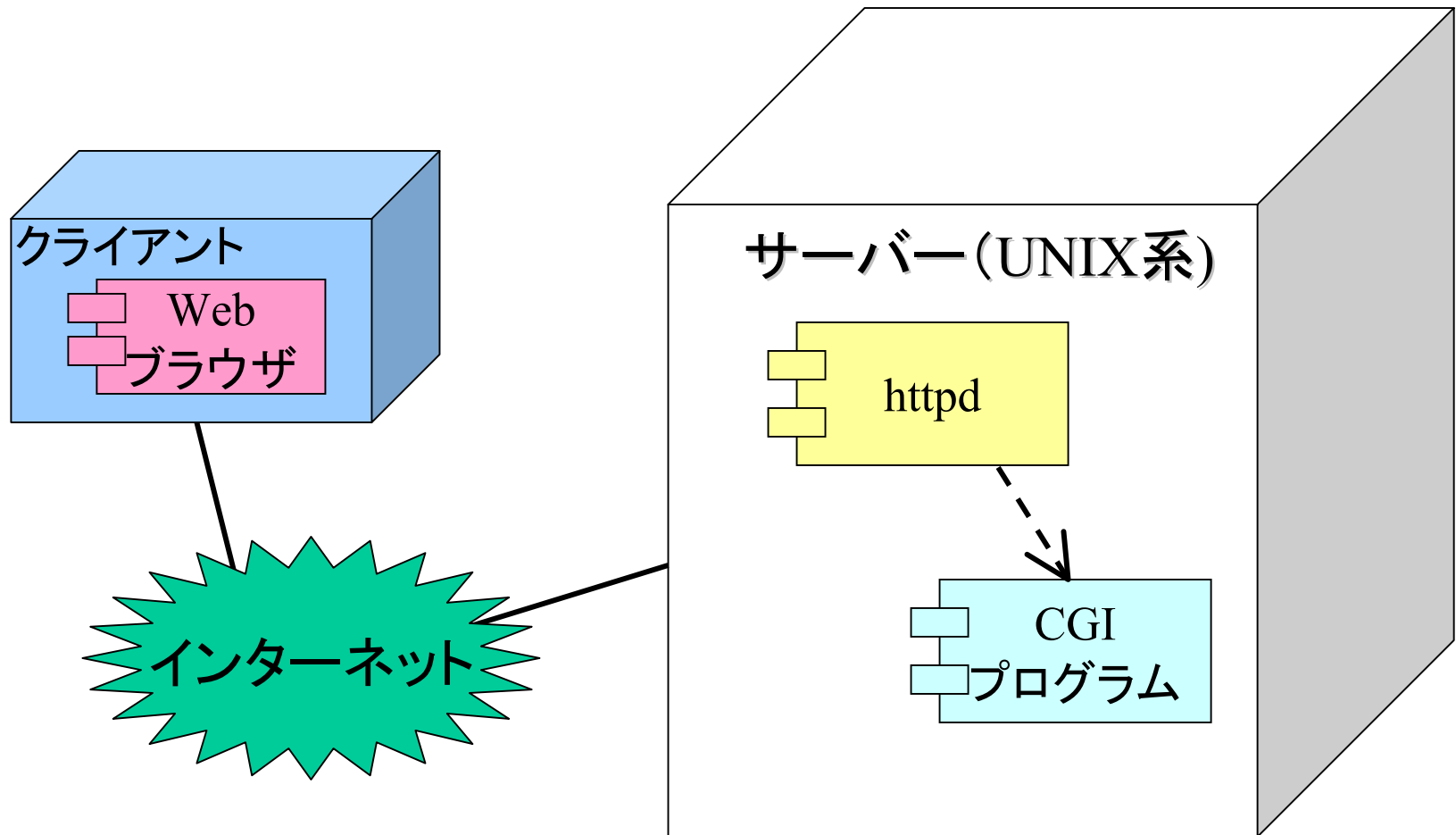
- いわゆるウェブサーバー.
- ホームページのデータを管理し, ブラウザからのリクエストがあれば, ブラウザにページのデータを送信するサービスを行う.
- httpdの種類や設定にもよるが, CGI等は, このhttpdの下請けとして他の(perl等の)プロセスが走る.

# httpdとクライアントの通信

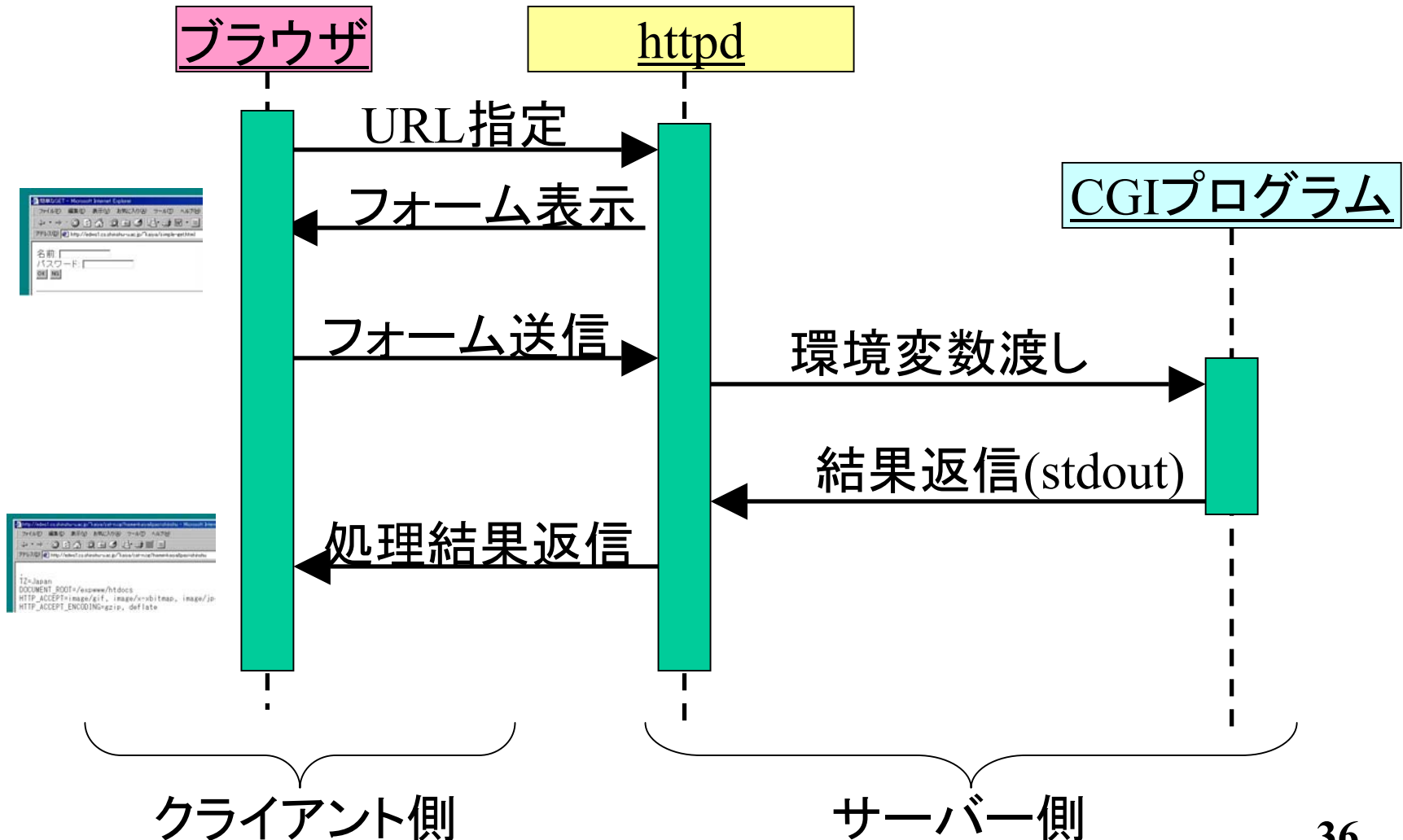
- 「ページをクリックする」に対応する内部的な命令
- ブラウザからサーバーに対して、**見せて欲しいページを注文**すること。



# システム構成 (UML風に)



# CGI実行のシーケンス例



# FreeWnn, canna

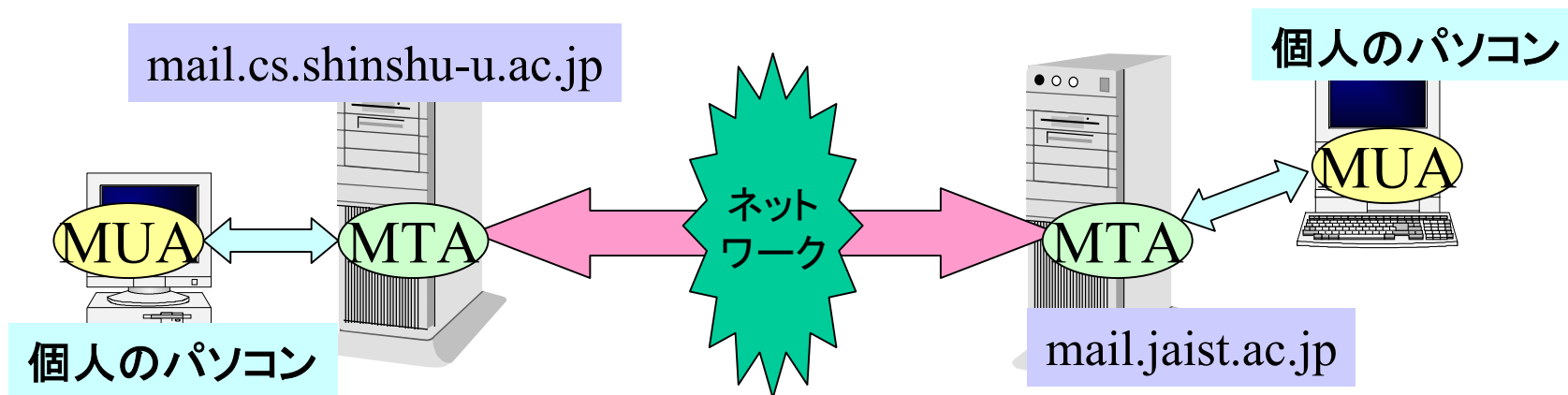
- 仮名漢字変換をするためのサービス.
- 伝統的にUNIX系OSでは, 仮名漢字変換をする処理をアプリ内ではなく, 一括してサーバーで管理する方式をとっている.
  - 理由は?. 昔, 計算機パワーが乏しかったため, 比較的早いマシンがこの処理を一括してやっていたため?
- Wnn, Cannaは仮名漢字変換サービスの中でも最もポピュラーなもの.

# sendmail, postfix

- MTA (Mail Transfer Agent) と言われる電子メールを送信するサービスを提供するデーモン.
- 皆さんのLinuxの場合, 自前のMTAを使わず, サーバー(mail.cs等)のMTAに接続してメール送信している.
- 受信はまた別.

# 補足

- MUA: メールユーザーエージェント  
OEなどを含むパソコンのメールクライアント
- MTA: メールトランスファーエージェント  
実際にメールの送信受信を行うサーバー  
sendmail, qmail など.



# smb

- sambaと言われるサービスの提供デーモン.
- LinuxのDiskやPrinterをWindowsからも利用できるようなサービスを提供している.



# Windowsの場合は？

イメージ名	ユーザー名	C...	メモリ...
VD.EXE	kaiya	00	5,588 K
XFR.EXE	SYSTEM	00	2,864 K
SVCHOST.EXE	SYSTEM	00	3,928 K
AOM.exe	kaiya	00	4,896 K
CONIME.EXE	kaiya	00	3,928 K
ApntEx.exe	kaiya	00	3,456 K
RTVSCAN.EXE	SYSTEM	00	12,340 K
PDS.EXE	SYSTEM	00	2,832 K
DEFWATCH.EXE	SYSTEM	00	2,460 K
Avsynmgr.exe	SYSTEM	00	3,444 K
ALG.EXE	LOCAL SERVICE	00	3,960 K
AcroTray.exe	kaiya	00	3,172 K
taskmgr.exe	kaiya	05	6,808 K
MSMSG.S.EXE	kaiya	00	5,680 K
SPOOLSV.EXE	SYSTEM	00	8,620 K
Directcd.exe	kaiya	00	7,416 K
Altime.exe	kaiya	00	3,712 K
CTFMON.EXE	kaiya	00	4,316 K
VPTRAY.EXE	kaiya	00	4,700 K
SVCHOST.EXE	LOCAL SERVICE	00	4,344 K
Apoint.exe	kaiya	00	5,508 K
SVCHOST.EXE	NETWORK SERVICE	00	2,256 K
S3tray2.exe	kaiya	00	3,848 K
PCTSPK.EXE	kaiya	00	3,608 K
S3hotkey.exe	kaiya	00	2,944 K
SVCHOST.EXE	SYSTEM	00	24,220 K
ttermpro.exe	kaiya	00	6,948 K
SVCHOST.EXE	SYSTEM	00	3,468 K
cmd.exe	kaiya	00	1,748 K
LSASS.EXE	SYSTEM	00	1,220 K
SERVICES.EXE	SYSTEM	00	12,664 K
WINLOGON.EXE	SYSTEM	00	600 K
CSRSS.EXE	SYSTEM	03	12,248 K
SMSS.EXE	SYSTEM	00	440 K
xvim.exe	kaiya	00	9,404 K
MSGSYS.EXE	SYSTEM	00	2,304 K
System	SYSTEM	00	256 K
System Idle Process	SYSTEM	82	20 K

- Linux同様, いくつかのデーモンが動いている.
- が, いつ, どのようなタイミングで起動しているかはわからん.