オペレーティングシステム

2022/11/15

海谷 治彦

目次

ネットワークの制御

- 1 OSとネットワーク
- 2 通信インタフェース: プロトコル
- 3 通信用プログラミングインタフェース
- 4 クライアント・サーバー
- 5 ネットワークを介したOS機能の利用
- 6 分散OS
- その他，ネットワーク系のOSサービス
- アンケート

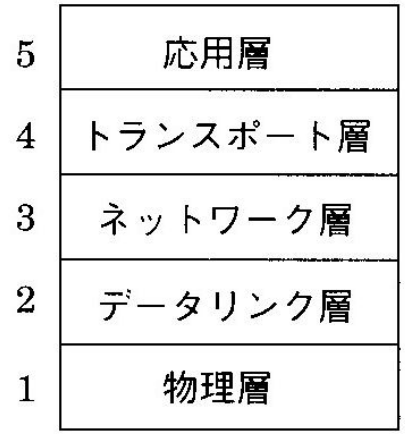
ネットワークの機構

- 詳細は「コンピューターネットワーク」の授業に譲るが・・・
- 単なる電気的，光学的な現象を寄り合わせ，意味ある情報交換を成立させているメカニズムである.
- 人間が単なる空気の振動を寄り合わせて，音，単語，言語，会話を成立させているのに似ている.

プロトコル Protocol

- 「通信規約」のこと.
- 通信する異なる者(物)同士が，予め送受信するデータの意味の合意をとっておくこと.
  - 人間でいえば，日本語，英語等の言語がプロトコルといえる.
- プロトコルは電気信号等の低レベルのものと，人間同士の会話のように意味のある高レベルのものとがある.

五階層モデル



OSIの7階層モデルの第5層と
第6層を除いたもの

図 11.1　ネットワークアーキテクチャの５階層モデル

- 低いプロトコルから高いプロトコルまで重ねたもの.
- 実体としては，物理層の電気的な信号しか無いが，
- それらを組み合わせることで，より意味のあるプトロコルを構成できる.

人間の会話との大雑把な対比

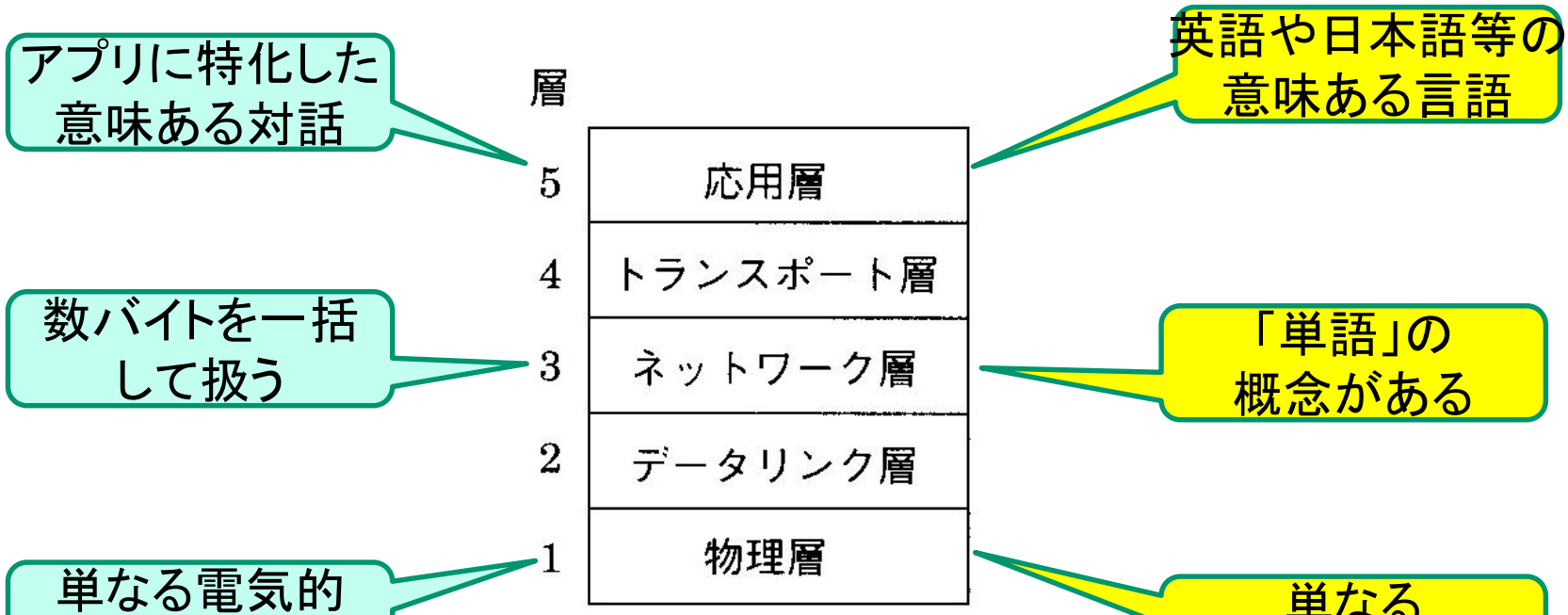
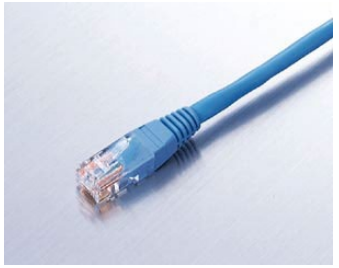


図 11.1　ネットワークアーキテクチャの５階層モデル

主な階層の具体的なプロトコル



- MAC (データリンク層)
  - イーサネット等で採用されている方式
- IP (ネットワーク層)
  - 20~60バイトのデータを一組(パケット)として送る.
  - パケットが事故で消える場合があっても再送等はしない.
- TCP (トランスポート層)
  - パケットの再送機構を含むため，データが必ず送り先に順番に届くことを保証できる.
- UDP (トランスポート層)
  - データの消失，重複，順番逆転がありうるが，多数の相手への同時送信(ブロードキャスト)ができる.
- HTTP (アプリケーション層)
  - ウエブブラウザ等が表示データの送受信に使うプトロコル.

通信のイメージ

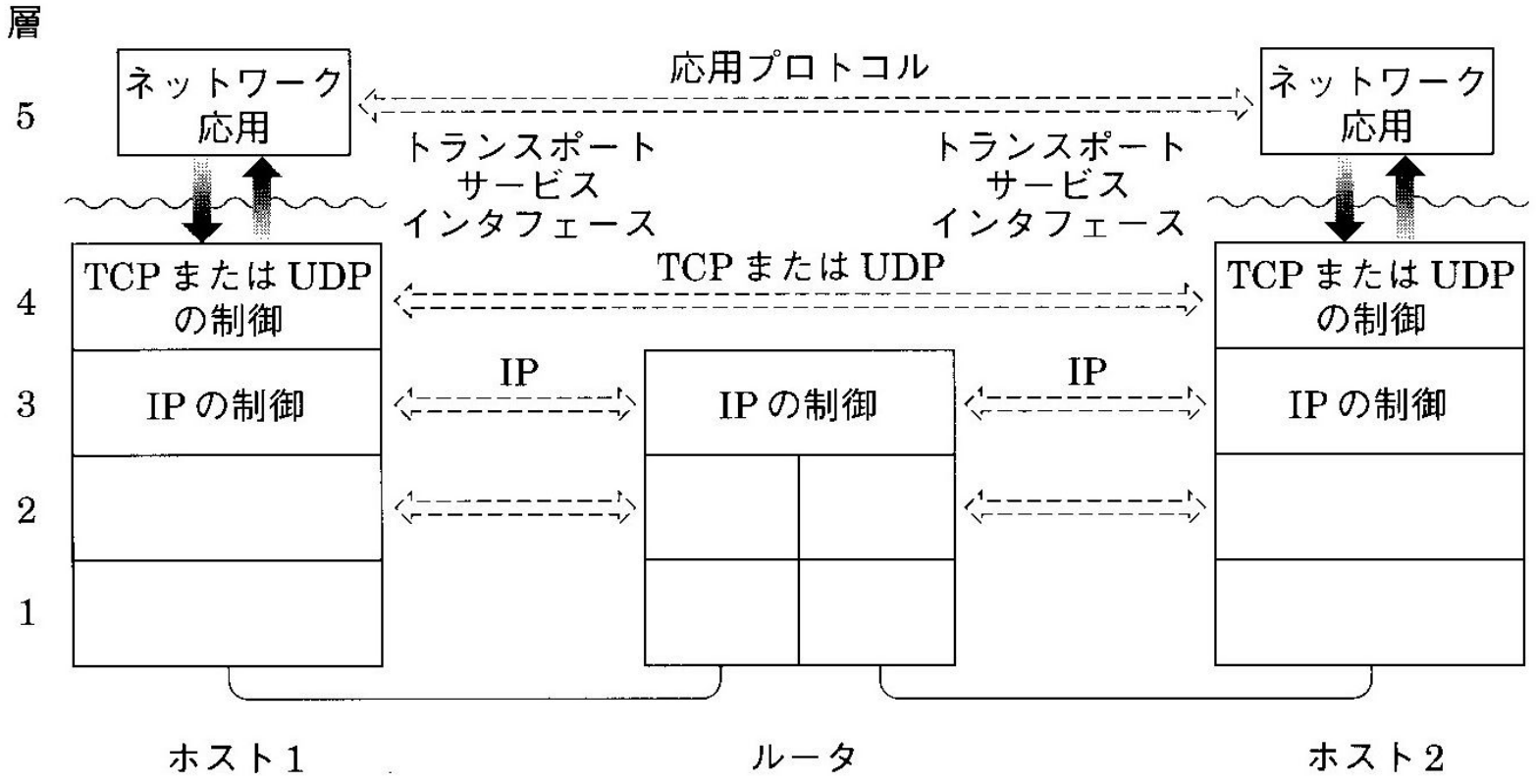


図 11.2　TCP，UDP，IP の位置付け

ネットワークでのOSの仕事

- 以下の二つに分類できる.
- 物理層での実データ交換を行っている機器(デバイス)からデータを受け取り，より上位のプロトコルを構成する仕事.
  - そもそもルーター等のネットワーク機器にもOSが搭載されている.
- 上位のプロトコルを用いて他の情報機器における情報資源にアクセスするための仕事.

OSから見たネットワーク

- ある意味, (ディスク等の)他の機器同様, アプリケーションのプロセスから容易に利活用できる準備をOSがする必要がある.
- そのためには,
  - 前述のプロトコル階層を理解する機構をOSに組み込む,
  - 各階層(主に上層)を操作するためのシステムコール関数群をOSがアプリに提供する.

  が必要となる.

プロセスから見たネットワーク

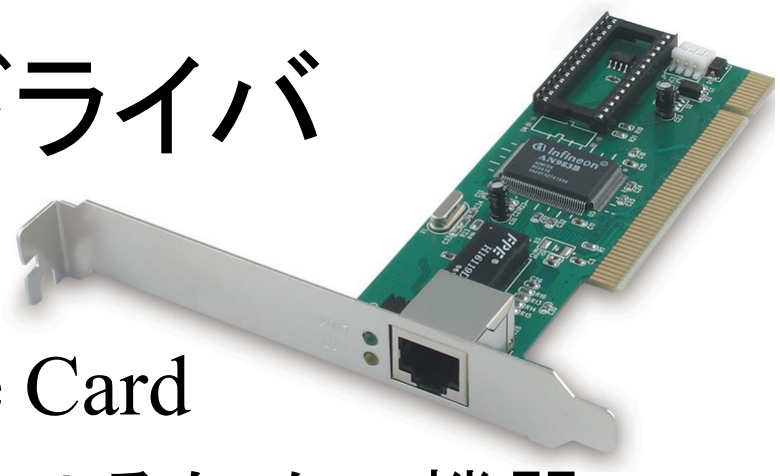
- プロセスの性質によって，どのレベルの層のシステムコールが必要か違ってくる.
  - 上層で良い場合: 例えばワープロ等はネットワークを意識する必要は無い.
  - 中～下層が必要な場合: ネットワークを監視するようなプロセスはIPもしくはそれ以下にアクセスするシステムコールが必要.

- 多様なプロセスがあるので，単に上層(より抽象化された層)だけのシステムコールだけでは困る.

OSのネットワーク機能

a. ネットワーク装置(NIC)のデバイスドライバとして機能
b. 論理的なネットワーク通信をシステムコールによってアプリが行うための支援機能
c. ネットワーク上もしくは他のコンピュータに接続された機器を利用するための機能
d. 他コンピュータに計算処理を依頼できる機能

a. NICのドライバ



- NIC: Network Interface Card
- 物理層の通信を成立させるための機器.
- いまどきのOSはこれらを操作するための デバイスドライバを持っている.
- 機器自体はデータリンク層までしか処理できない.
- ネットワーク層より上は，OS内のプログラム(デバイスドライバ)で実現されている.

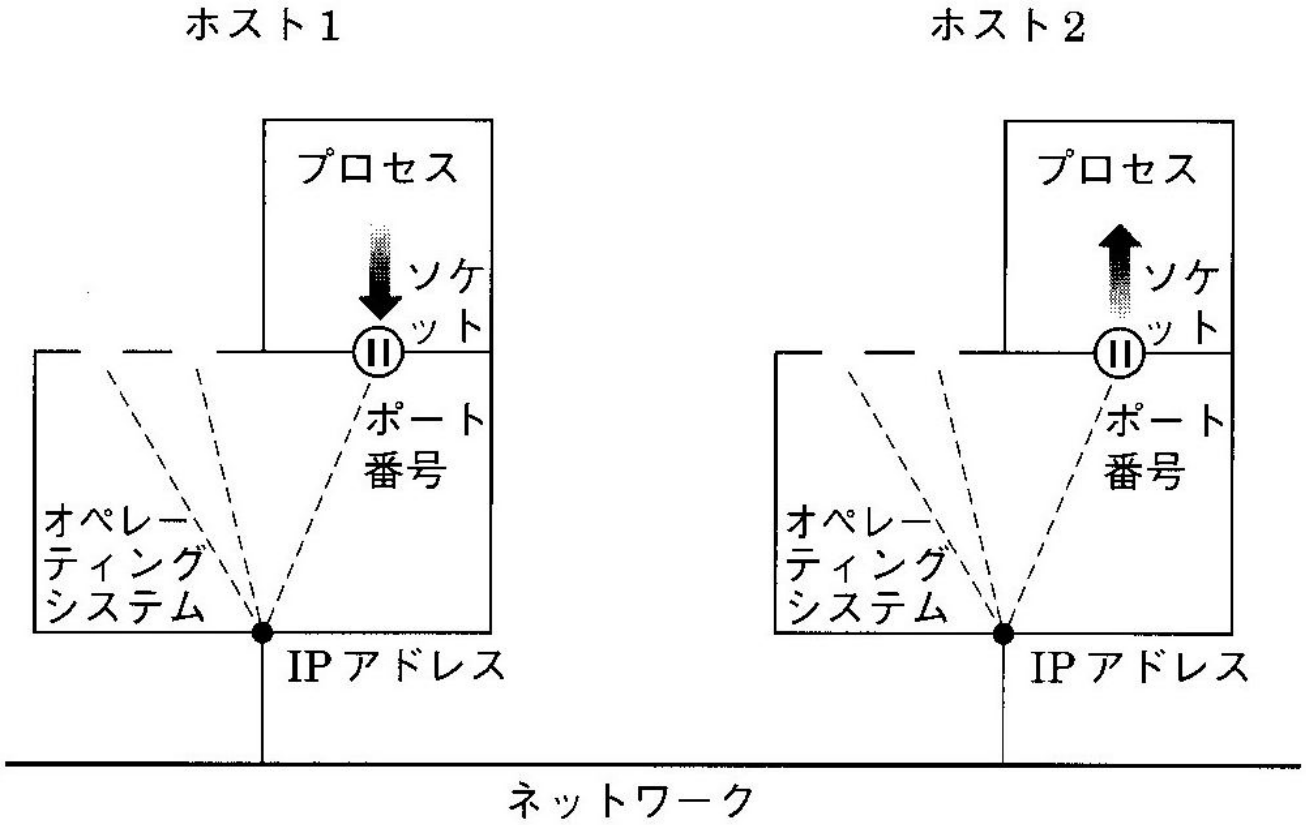
b. 論理的なネットワーク通信

- IP, TCP, UDPレベルの通信を行うためのシステムコールが実現されており，アプリケーションから利用可能となっている.
- 前述のようにTCPを使えば，データ送信の順序やデータが紛失していないことの保障がされる.
  - OSが内部的に再送依頼や順番調整をしてくれているため.
  - 詳細は後述.

論理的ネットワークの実現

- IPプトロコルに基づく通信が典型的な例.
- ネットワークのインタフェースにIPアドレスと呼ばれる値を割り振り,
- その値に基づき通信を行う
- 加えてポートと呼ばれる番号によって，複数種類の対話を同時に行うことができる
- コネクションの有無
  - コネクション型: 送信側と受信側の双方が通信を行っていることを意識している. TCPに対応.
  - コネクションレス: 送信側は受信側の準備完了の有無に関係なくデータを投げる. UDPに対応.

通信のイメージ



ホストはIPアドレスで区別される.

図 11.3　ソケットによる通信の概要

ソケット socket

- IP, TCP, UDPの通信プログラムを平易に行うための概念.
- 基本，電気のコンセント(socket)の比喩.
- より具体的には，システムコールを含むいくつかの関数群が提供されており，
- それらを組み合わせて通信プログラムを簡単？に作成できる仕組み.
- パイプの拡張概念と思ってよい.
  - パイプは同じマシン内のプロセス間に有効，ソケットは異なるマシン上のプロセス間でつなげられる.
- サンプルプログラムあります
  - 少なくとも server1.c client1.c は動作確認した.

Socket通信のイメージ

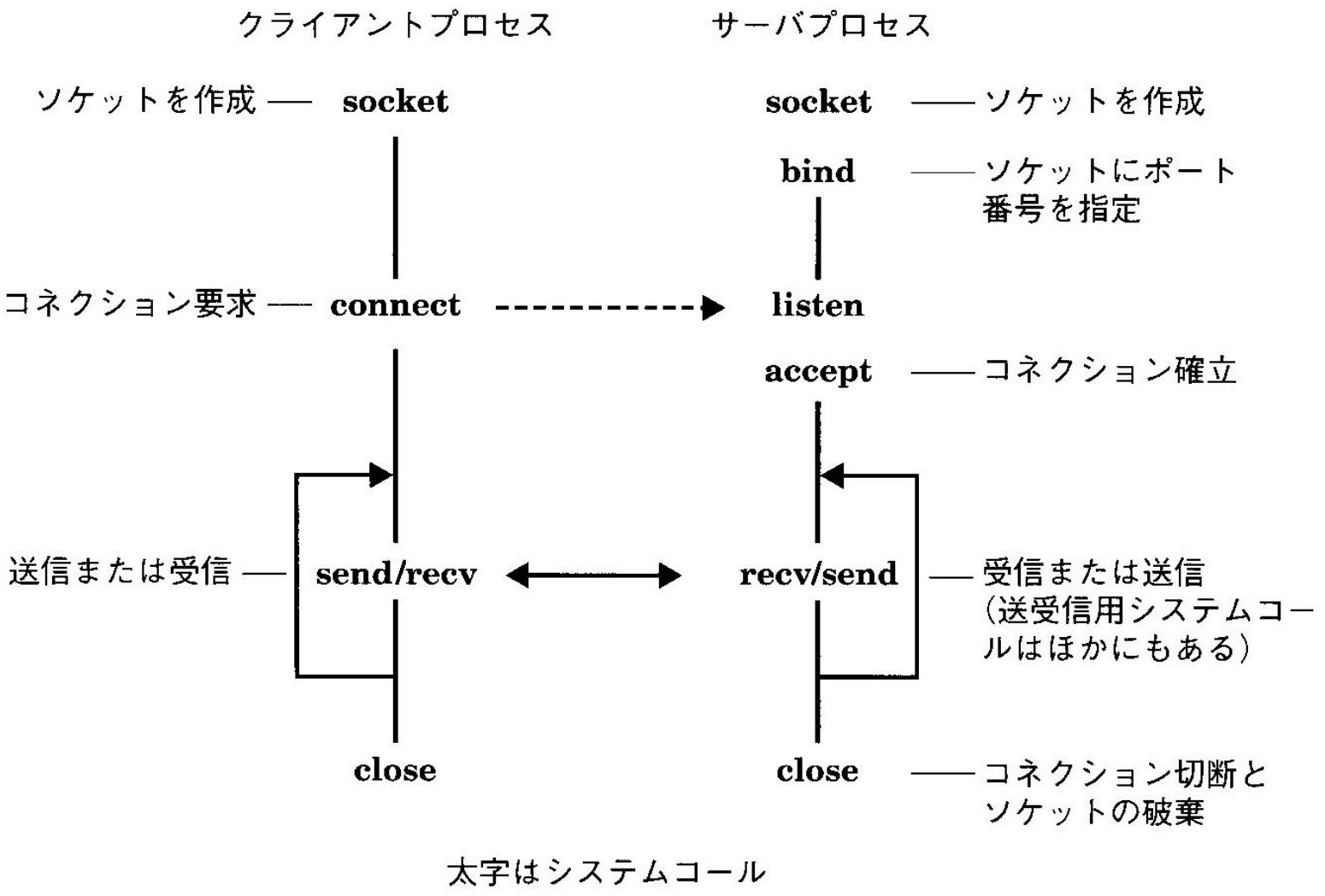


図 11.4　ソケットを用いた通信のプログラムの流れ（コネクション型の場合）

c. 外部機器の利用

- ファイルシステム，プリンタ，スキャナ等の比較的遅めの機器を，ネットワーク上の他のコンピュータに使わせてあげる機能をOSは持つ.
- メモリやCPU等，高速な機器の共用は研究的なレベル以外では行われない.

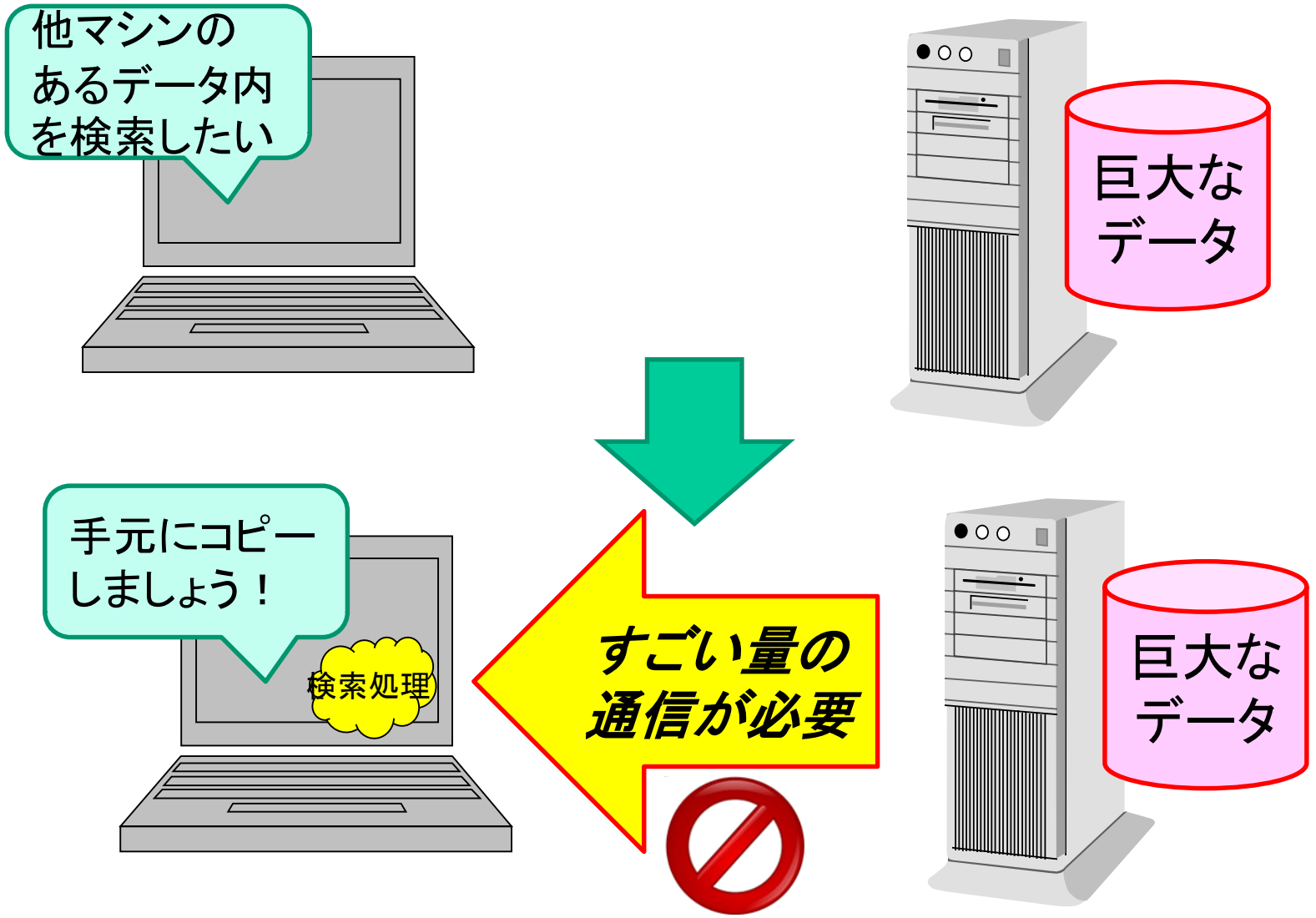
外部機器利用の実現

- ファイルシステムの共有
  - 古くからNFSと呼ばれるファイル共有プログラムがUNIX系OSには実装されている.
  - Windows系でも同様のものがある.
  - dropbox等も，同社が持つサーバーOSのファイルシステムを共有しているといえる.
  - NASと呼ばれる専用機器も売ってますよね.
- プリンタの共有
  - UNIX系では古くはlpd昨今ではcupsdと呼ばれるデーモン(後述)がプリンタの共有補助をしている.
  - Winでも同様のものがある.
  - プリンタも昨今は単独でネットにつながってますよね.

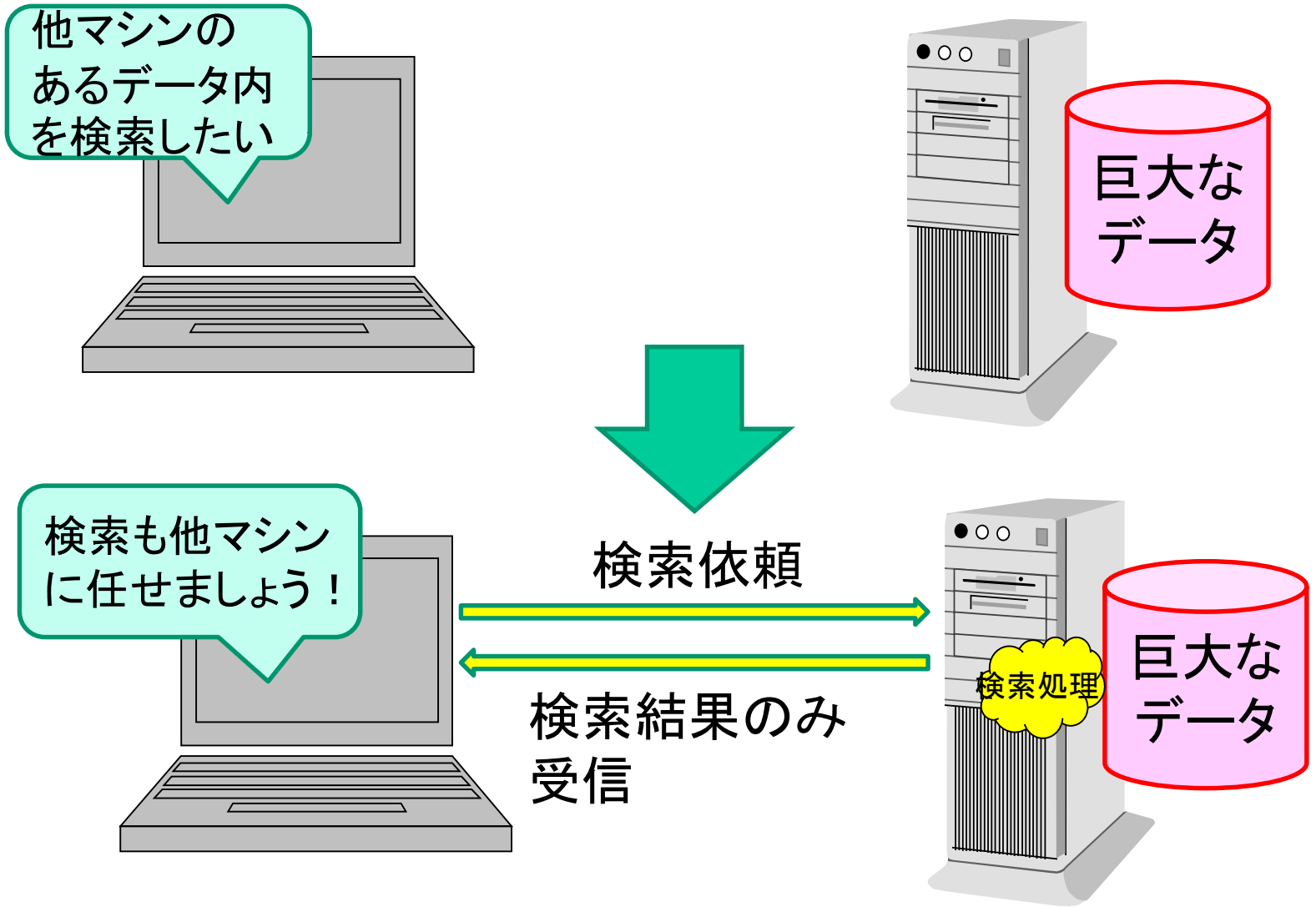
d. 他コンピュータへの計算依頼

- アプリケーション内の関数呼び出しレベルで，他のマシンへ計算依頼することを可能としているOSもある.
  - 通信をほとんど意識しないで済む.
- 通信を，より抽象化してサービス呼び出しと見なしている.
- 例えば外部のデータベース検索をデータベースを管理するマシンに依頼する等.
  - データベースを一旦手元のマシンにコピーして処理していたのでは大変過ぎる
- いわゆる「マッシュアップ」mashup と呼ばれるプログラミングを可能とする.

他への計算依頼が必要な説明 1/2



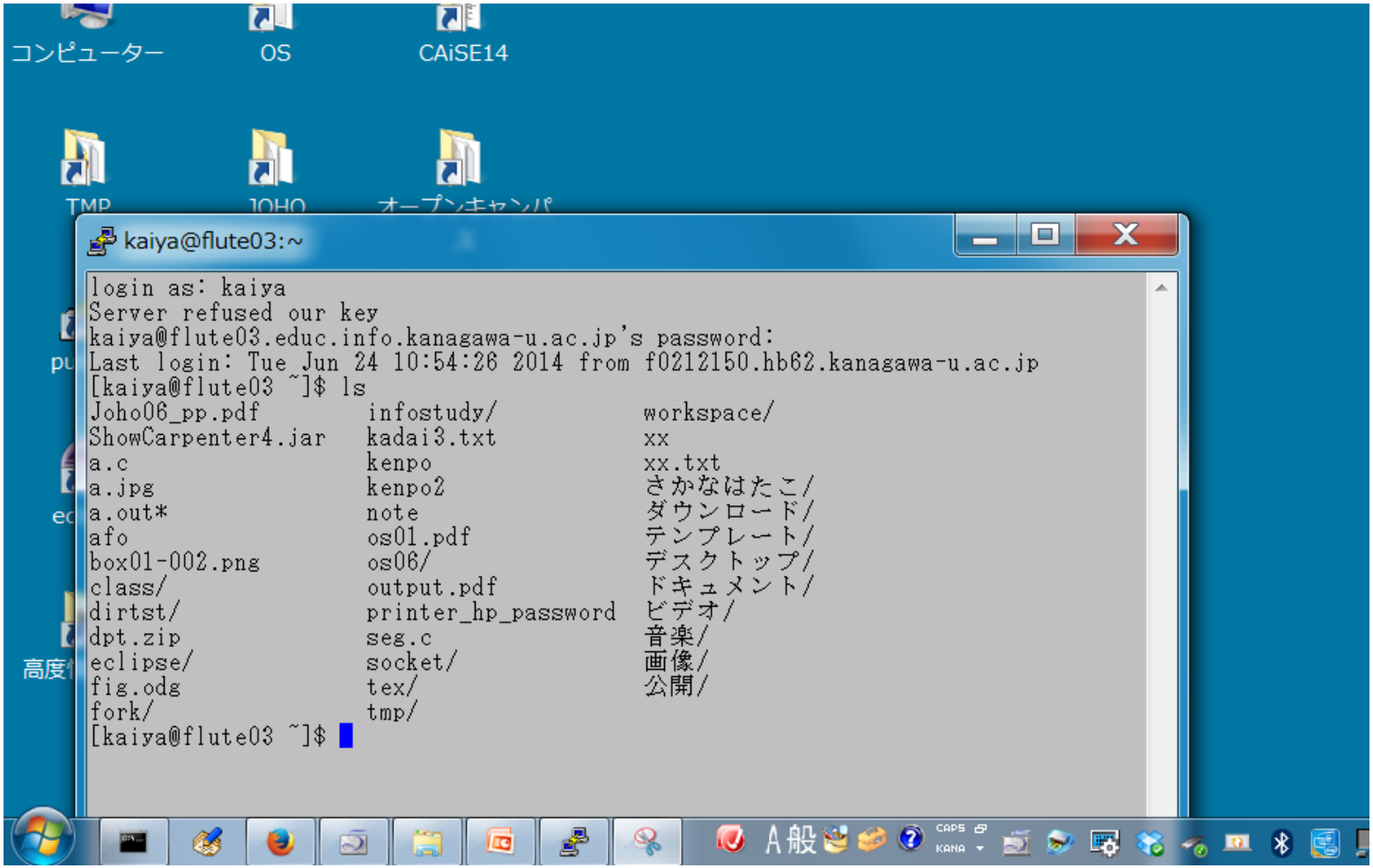
他への計算依頼が必要な説明 2/2



他コンピュータへの依頼実現

- リモート利用
  - テキストで入出力を行うプログラムを中心に利用することが可能.
  - 他のマシンのshellを呼び出し操作することができる.
    - WindowsからUNIXのshellを呼び出すことができる等.
  - telnet, rsh, ssh 等，リモート利用のためのコマンドがある．前者2つは安全でないので昨今は使われない.

sshの例

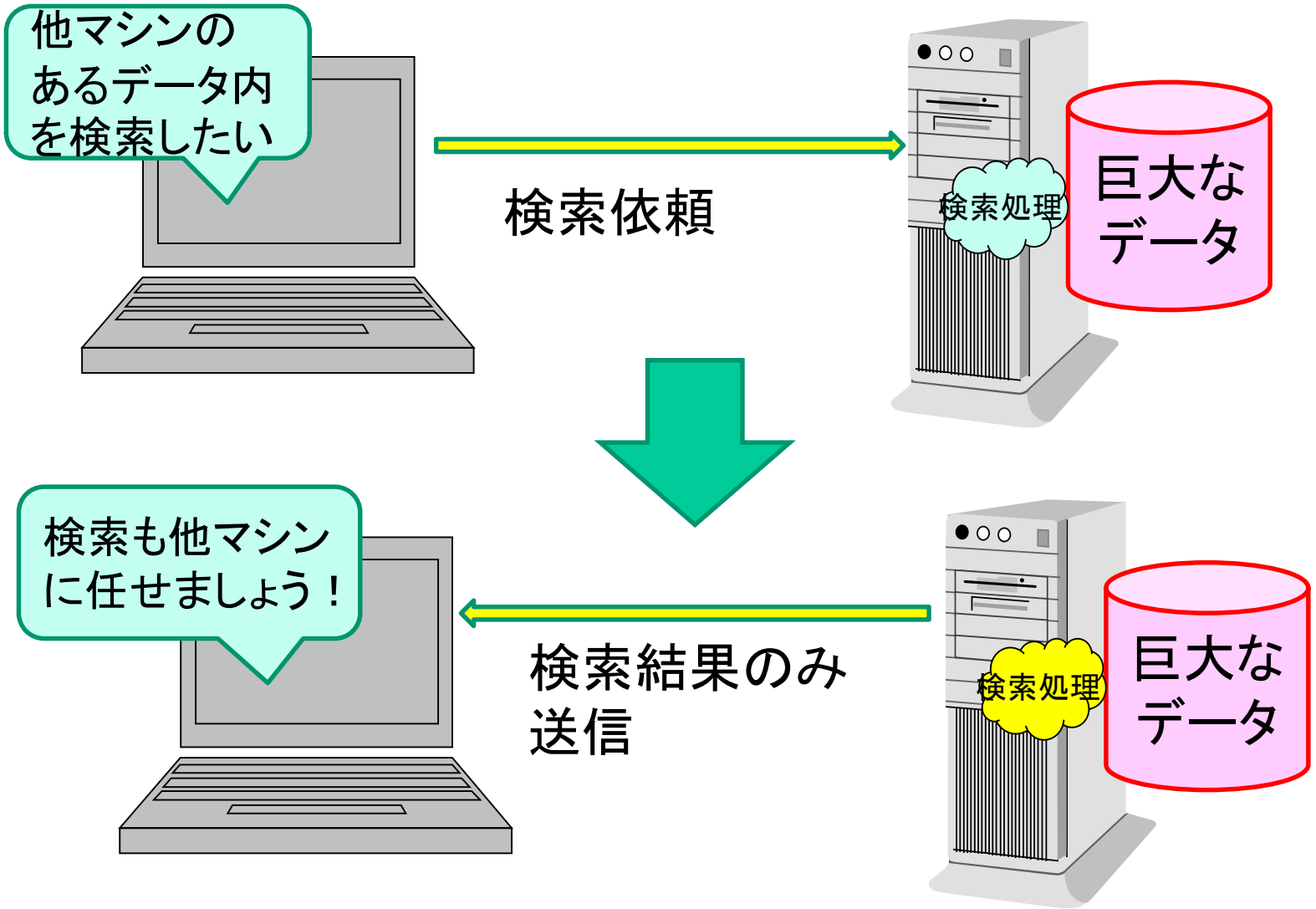


Win7マシンからLinux(2-101のマシン)にsshしている.

リモート手続き呼び出し

- プログラム内において，関数レベルで，他のマシンに処理を依頼(委譲)することができる.
- ただし，他のマシン側で処理依頼を受け付ける準備ができている場合.
  - 原則，受付のためのプロセスが常時動いている必要がある.
- 以下が代表的
  - RPC: Remote Procedure Call. 現在のUNIX系OSでも広く使われている．30年近くの歴史がある.
  - CORBA: 教科書では念入りに説明しているが今は廃れているとしか言えない.
    - Java/RMI も微妙に廃れている

RPCの基本的な考え方



RPCのメカニズム

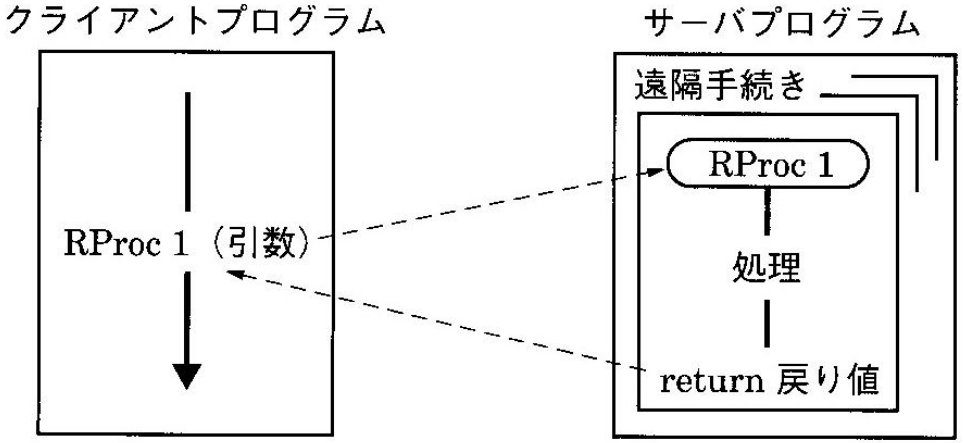


図 11.6　遠隔手続き呼出し（RPC）

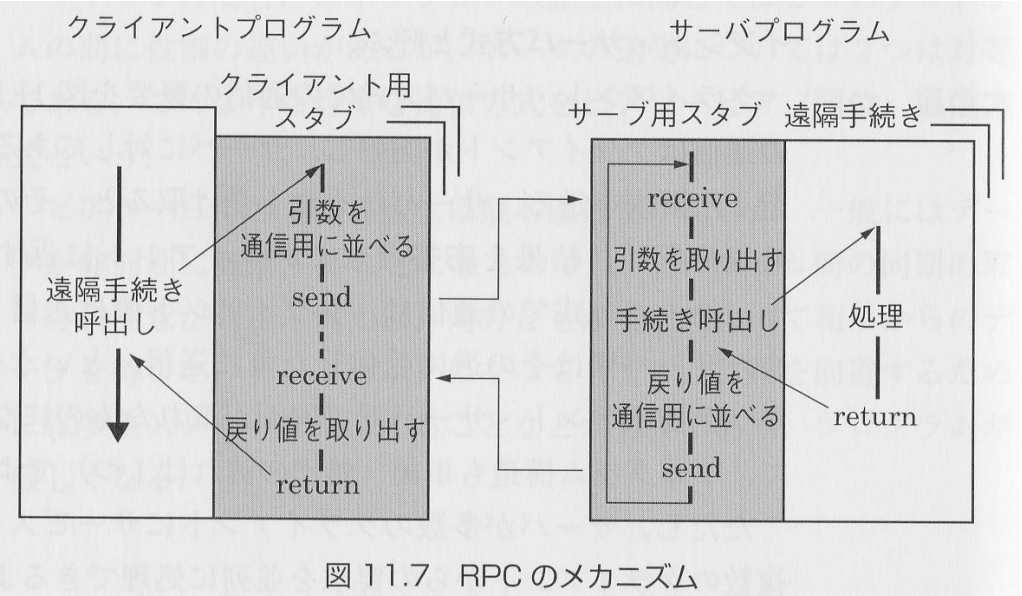


図 11.7　RPC のメカニズム

RPCとCORBAの大きな違い

- CORBAは仕様が複雑なため使うのが面倒.
  - 呼び出しを行うにもIDLという割と面倒な言語を使わないといけない.
- CORBAには，他のマシンがどのような関数呼び出し(サービス)を受け付けているかの電話帳である ORB (Object Request Broker)がある.
  - プログラム内でORBで利用可能なサービスを調べて関数を呼び出す等の便利がことができる.
  - 一方，利用可能なサービスが充実してないと残念なことになる.
    - 結果として大して普及しなかったのかも.

CORBAが流行らなかった理由

- 仕様が複雑で準拠していたプログラムを保守するのが大変だった.
- その結果として，シンプルなものが残っている.
  - その意味ではRPCはわりとシンプル.
- 現在はXMLベースのWebサービスを通りこして，json等のより簡易なデータに基づくWebサービスが主流のようだ.
  - いわゆるマイクロサービスというやつ.

CORBAの複雑さ

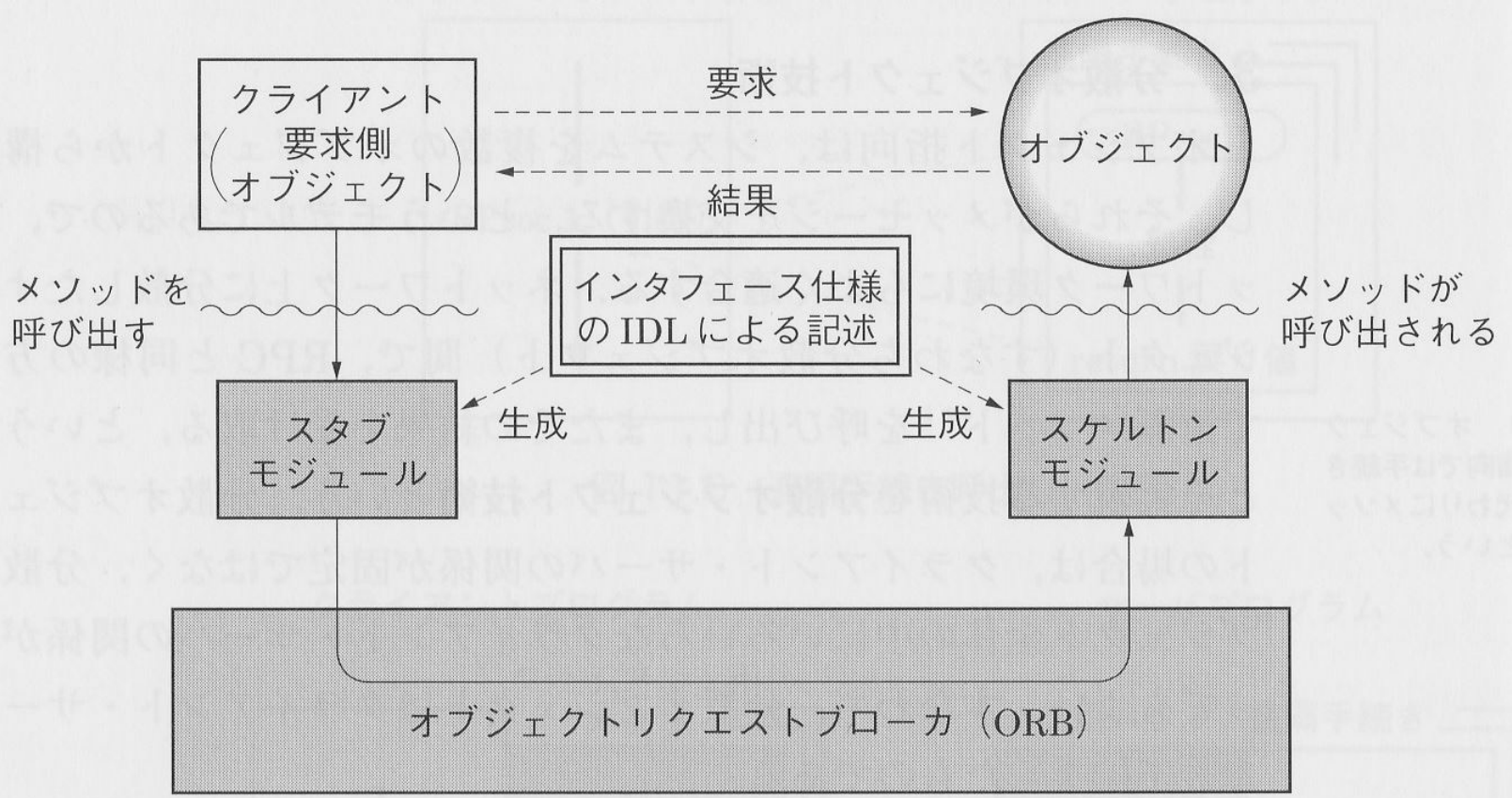


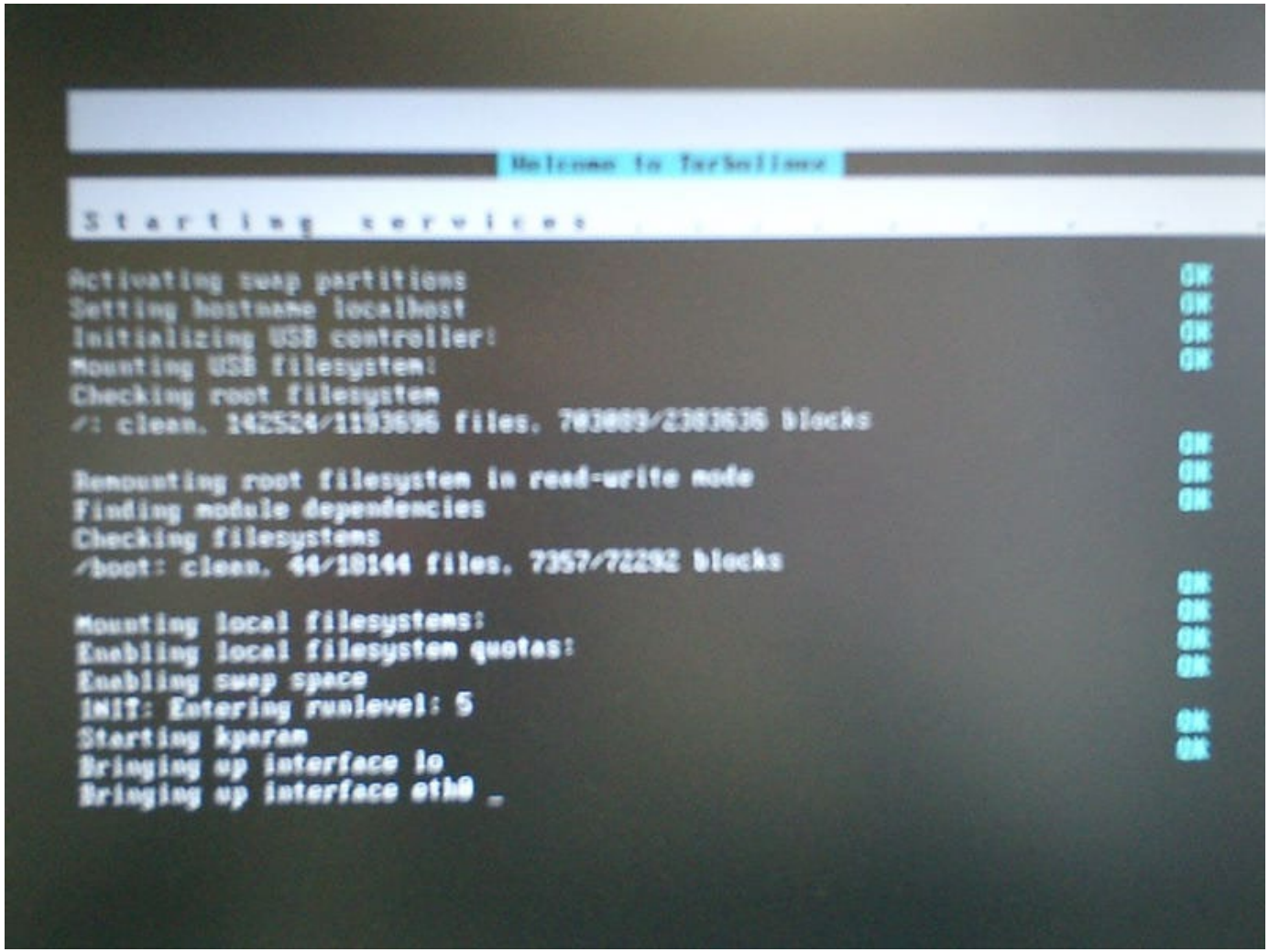
図 11.8　CORBA による通信のメカニズム

ネットワークを支えるプロセス群

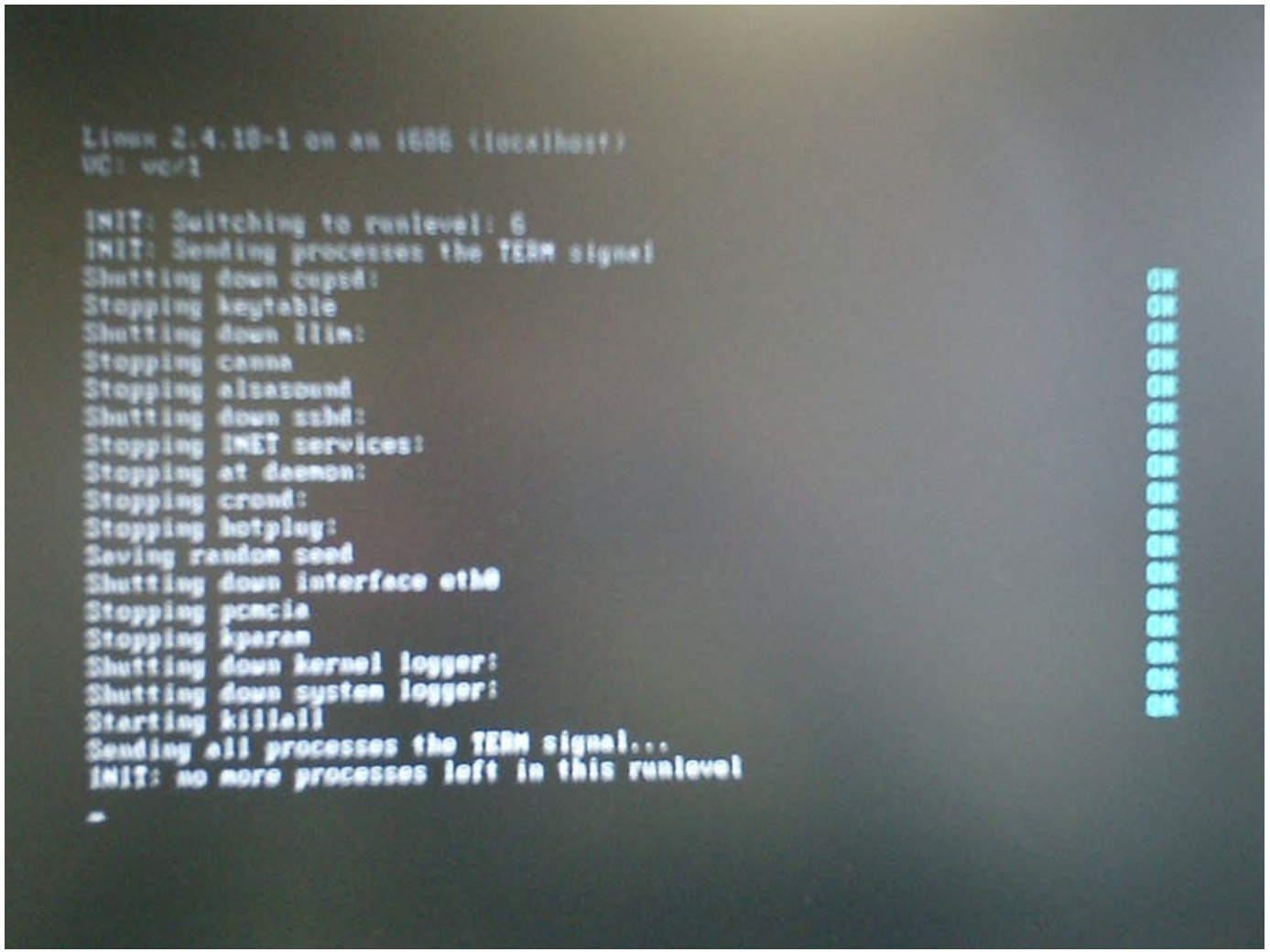
- OSのカーネルを補助する形で，数多くのプロセス群がOS起動時に実行され，常時動いている.
- このようなプロセス群のことをデーモンと呼ぶ.
- デーモンの一部はネットワーク関係の補助を行っている.

- LinuxもWindowsも本来，起動時に起動するデーモンの表示をしているが，昨今，素人向けのため，この表示を行わない.
  - とある設定ファイルをいじると表示されるようになる.

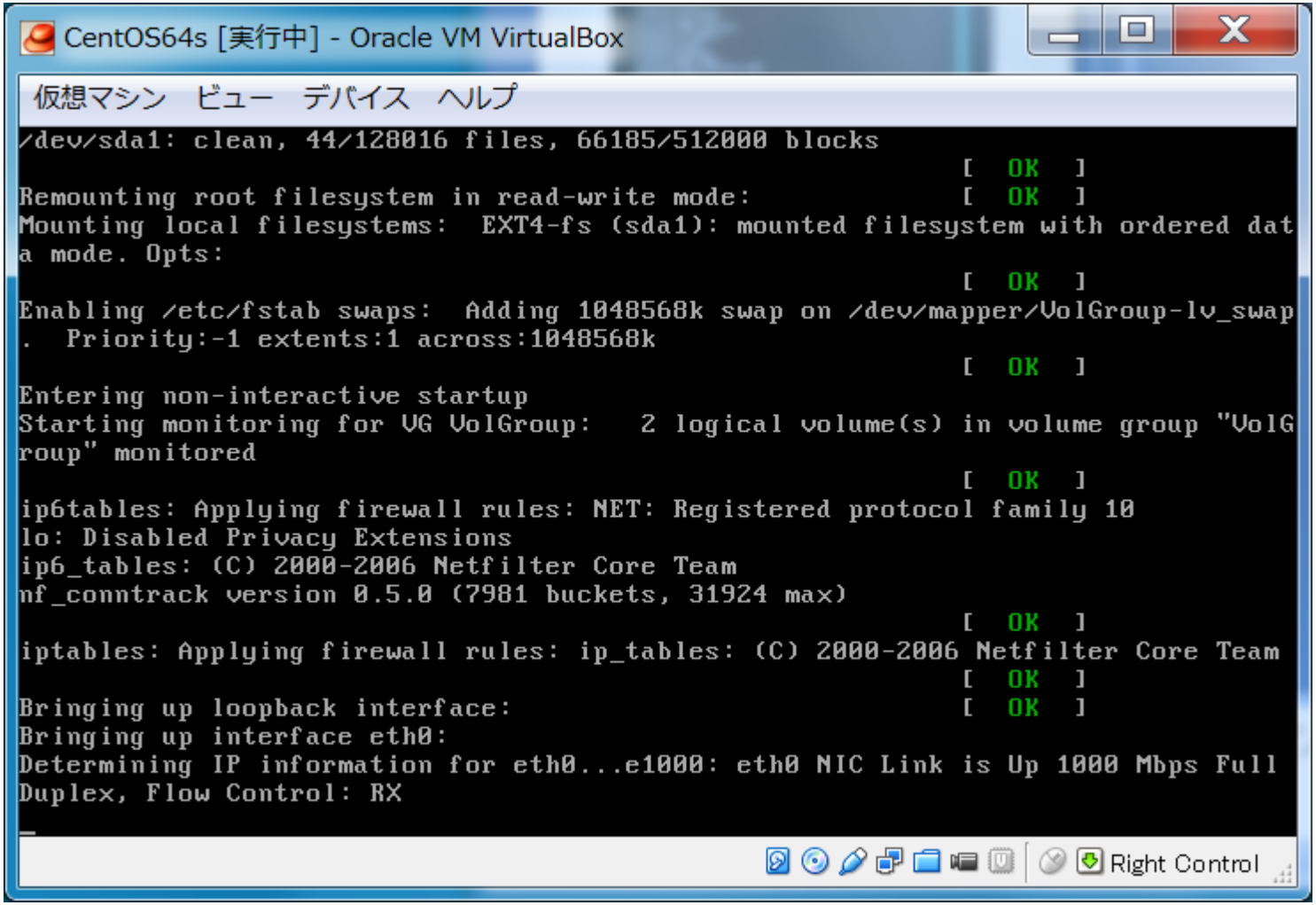
Linuxを起動すると・・・



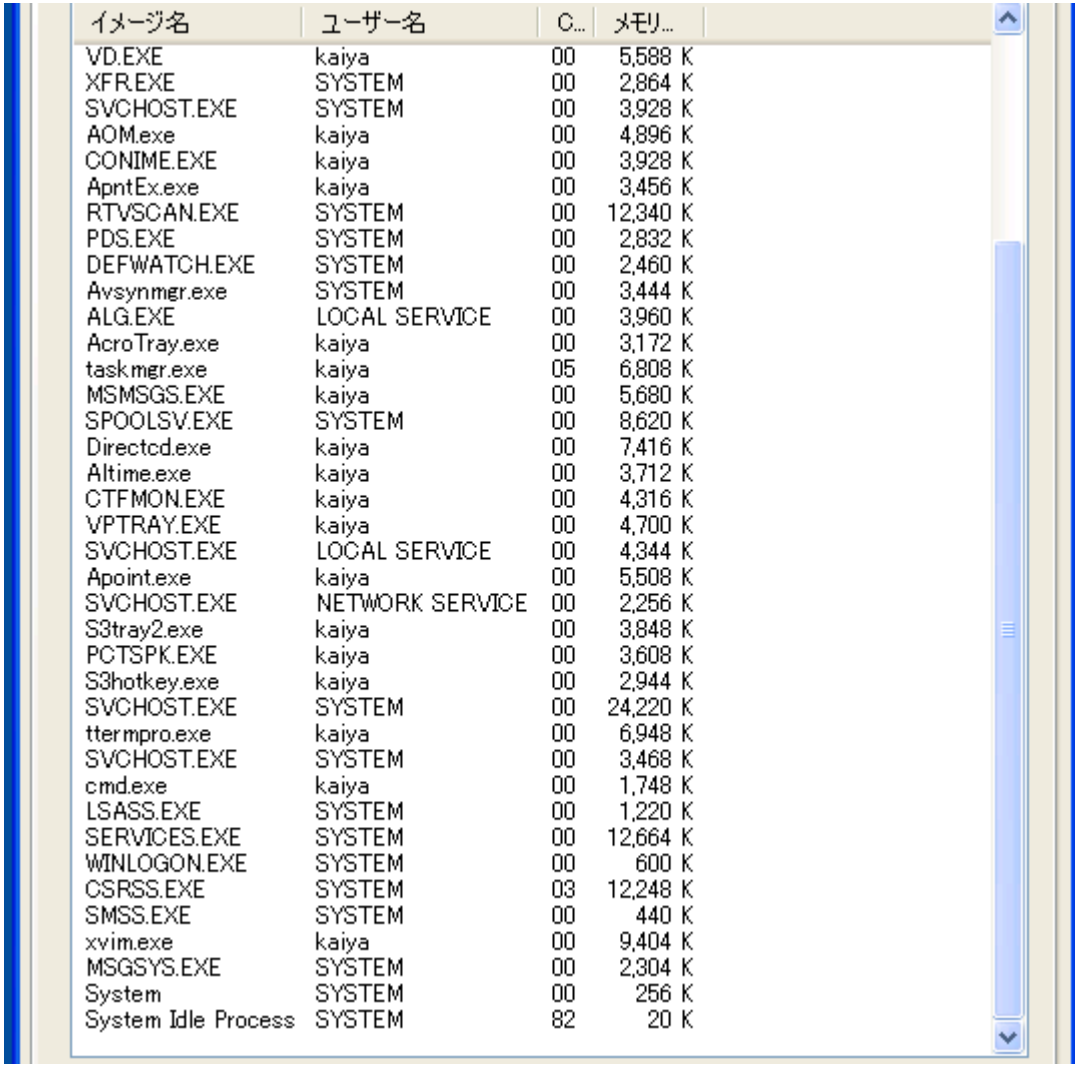
Linuxを停止すると・・・



最近の画面


CentOS64s [実行中] - Oracle VM VirtualBox
仮想マシン ビュー デバイス ヘルプ
/dev/sda1: clean, 44/128016 files, 66185/512000 blocks
[OK]
Remounting root filesystem in read-write mode: [OK]
Mounting local filesystems: EXT4-fs (sda1): mounted filesystem with ordered dat
a mode. Opts:
[OK]
Enabling /etc/fstab swaps: Adding 1048568k swap on /dev/mapper/VolGroup-lv_swap
. Priority:-1 extents:1 across:1048568k
[OK]
Entering non-interactive startup
Starting monitoring for VG VolGroup: 2 logical volume(s) in volume group "VolG
roup" monitored
[OK]
ip6tables: Applying firewall rules: NET: Registered protocol family 10
lo: Disabled Privacy Extensions
ip6_tables: (C) 2000-2006 Netfilter Core Team
nf_conntrack version 0.5.0 (7981 buckets, 31924 max)
[OK]
iptables: Applying firewall rules: ip_tables: (C) 2000-2006 Netfilter Core Team
[OK]
Bringing up loopback interface: [OK]
Bringing up interface eth0:
Determining IP information for eth0...e1000: eth0 NIC Link is Up 1000 Mbps Full
Duplex, Flow Control: RX
Right Control

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
MSMSGS.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，どういうタイミングで起動しているかはわからん．

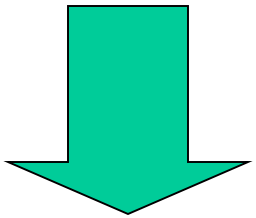
復習

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われない.

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

復習(os2より)

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

l0:0:wait:/etc/rc.d/rc 0
以後略
```

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

- 各runlebel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とはちょっと違う.

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.
  - コマンドを組み合わせた簡易プログラム
  - Windowsのバッチファイルに近い
- 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

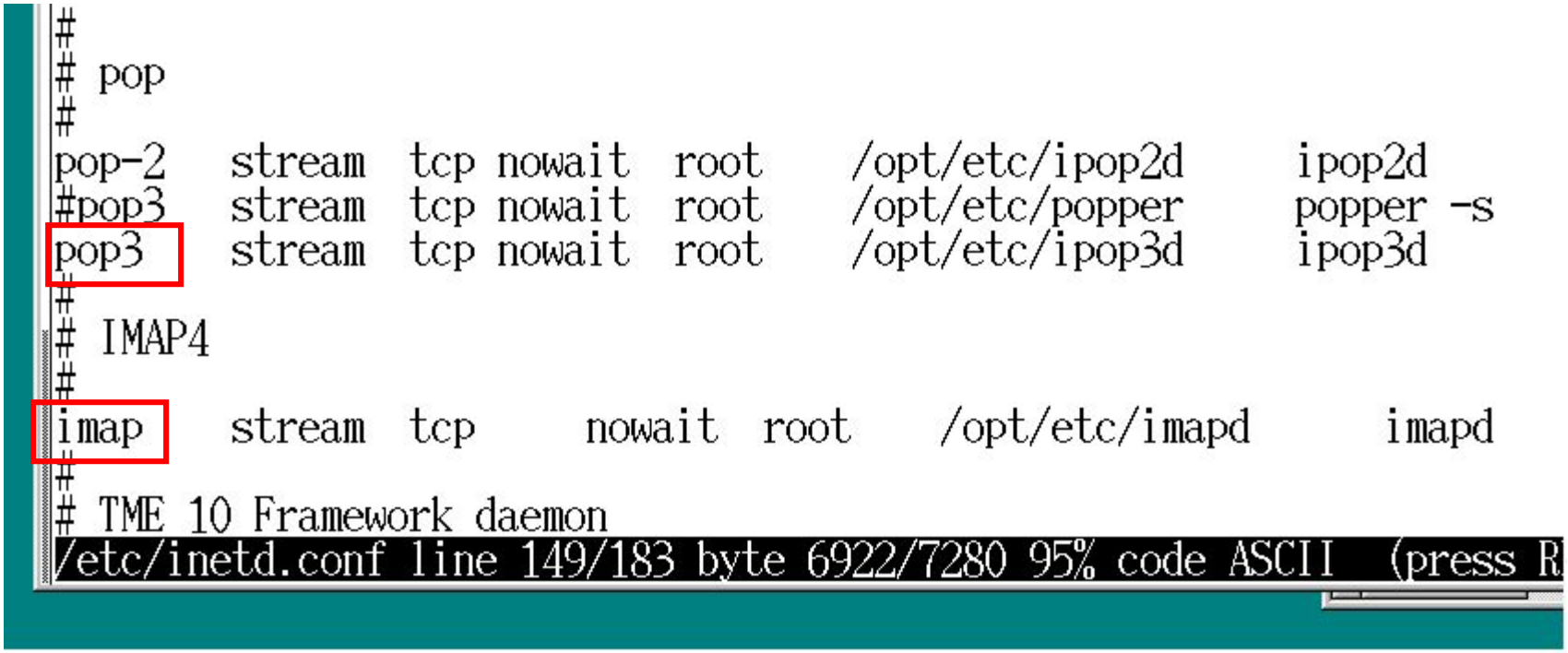
- 他のマシンからの手続き呼び出し(RPC)の設定補助デーモン.
  - RPCはマシンの中で番号付けされて管理されており，これをポート番号と言う.
    - 有名なサービスには共通の番号を使うように推奨されている.
      - wwwは80番，メール送信が25番等
  - portmapは，あるRPCにポートと対応付ける(mapする)ことで，通信ができるように補助してくれるサービス.

inet

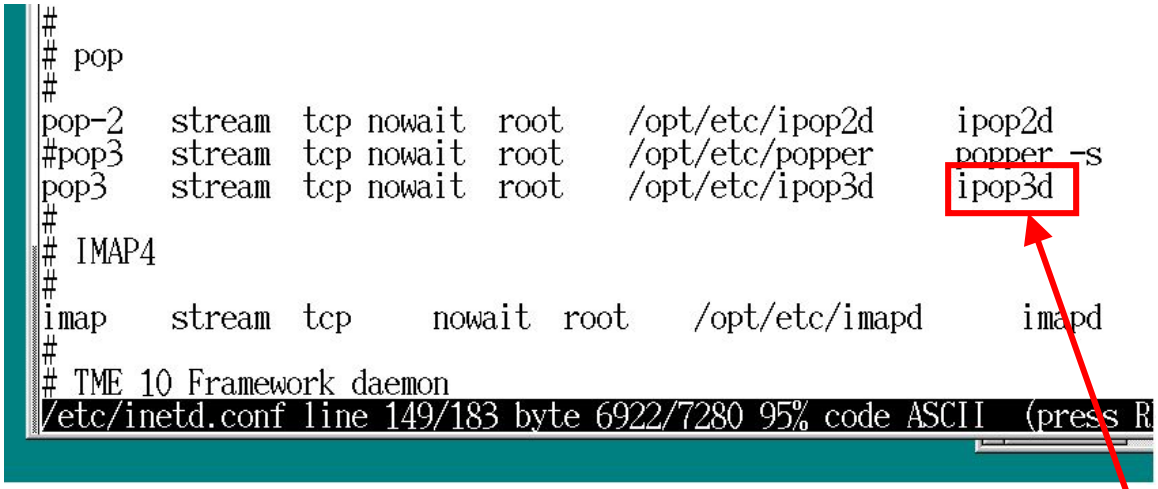
- 他のマシンからの要求に応じて，任意のサービスをオンデマンドで起動するためのデーモン.
  - 動作中は全ての通信内容を観察し,
  - あるポートに新しい接続要求が入ると,
  - その要求に応じたデーモンを起動して,
  - そのポートの処理要求を起動したデーモンに委譲する.
  - /etc/inetd.conf に設定がある.
- あまりセキュアでないので，最近は使われない.

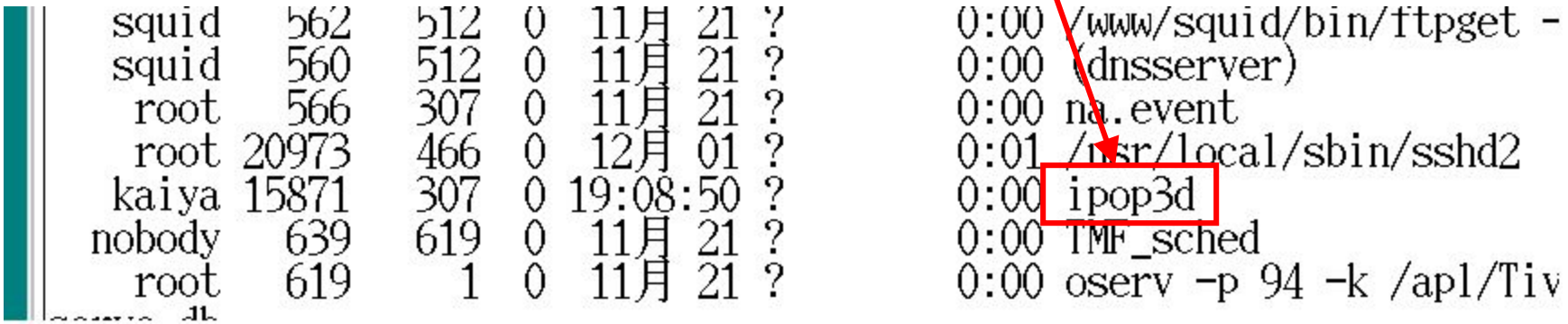
POPの例

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



POPでメールを読んでいる状態





リモートログインの例

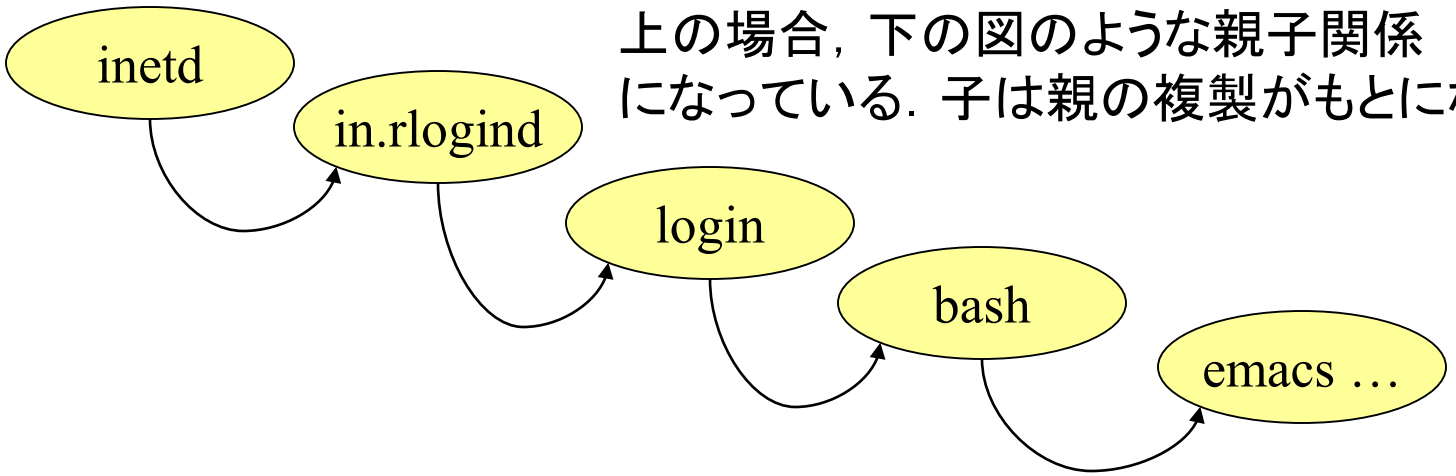
```
UID          PID  PPID  C STIME TTY          TIME CMD
root         538     1  0 Aug27 ?        00:00:00 inetd
root         983   538  0 23:22 ?        00:00:00 in.rlogind
root         984   983  0 23:22 pts/1    00:00:00 login -- kaiya
kaiya        985   984  0 23:22 pts/1    00:00:00 -bash
kaiya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

プロセスのもとになったコマンド名



上の場合，下の図のような親子関係になっている．子は親の複製がもとになっている．

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.
- 最近はコレのかわりに，capsd というのが使われる.

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.

cron

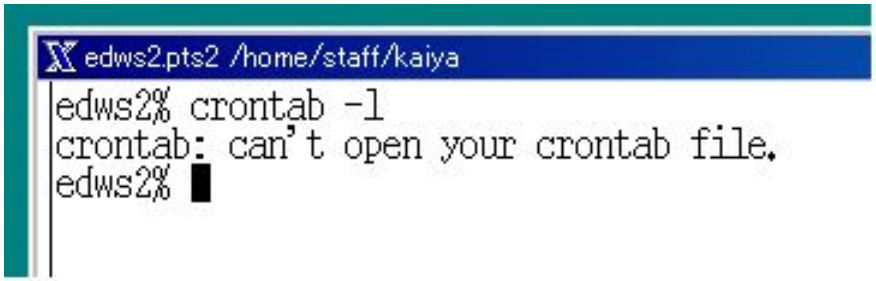
- 定期的にコマンドを実行するためのシステム.
- ユーザー別に設定される.
- UNIX系には大抵ある.
- Winでいう所の「タスク」に相当.

- 詳しくは man cron を参照

実行される内容の確認・記述

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.



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

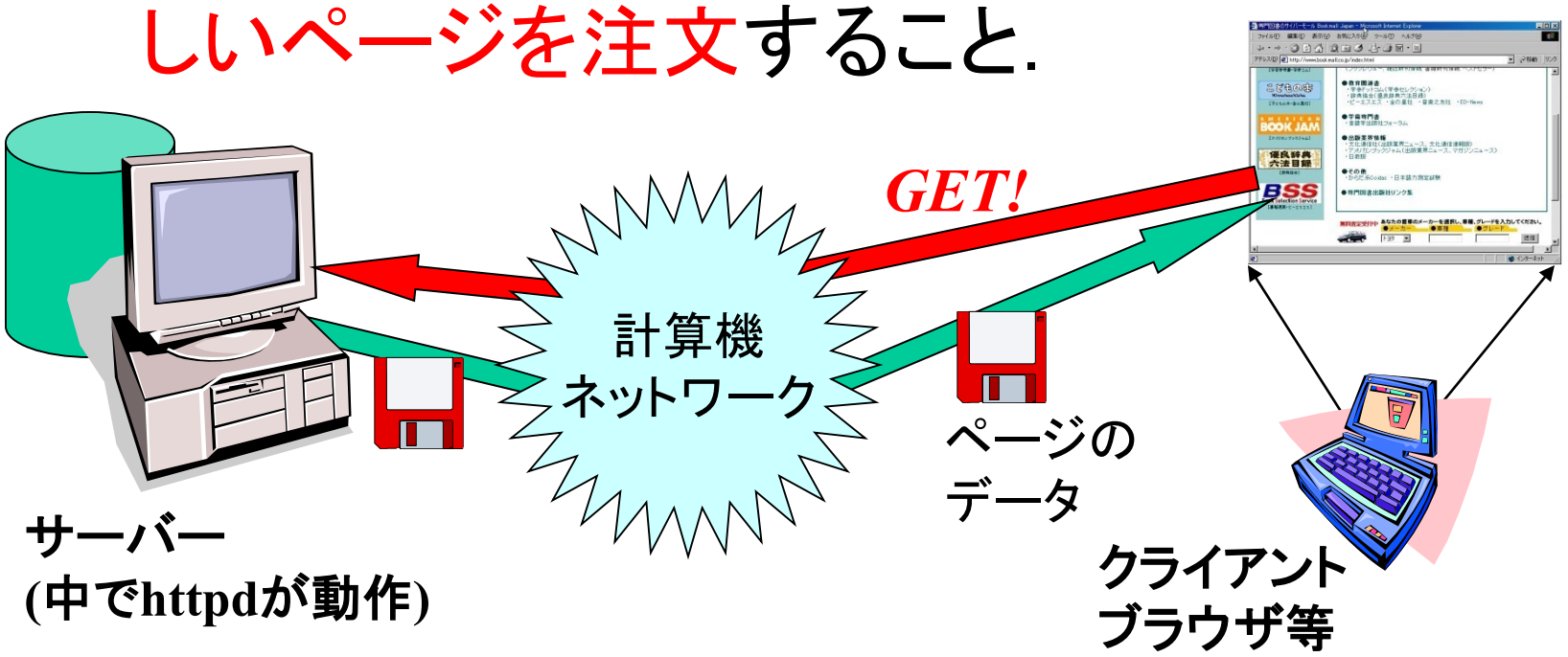
- 暗号化された安全なリモートログインをサービスするデーモン.
- 昔のrshdやrlogindの代わり.

httpd

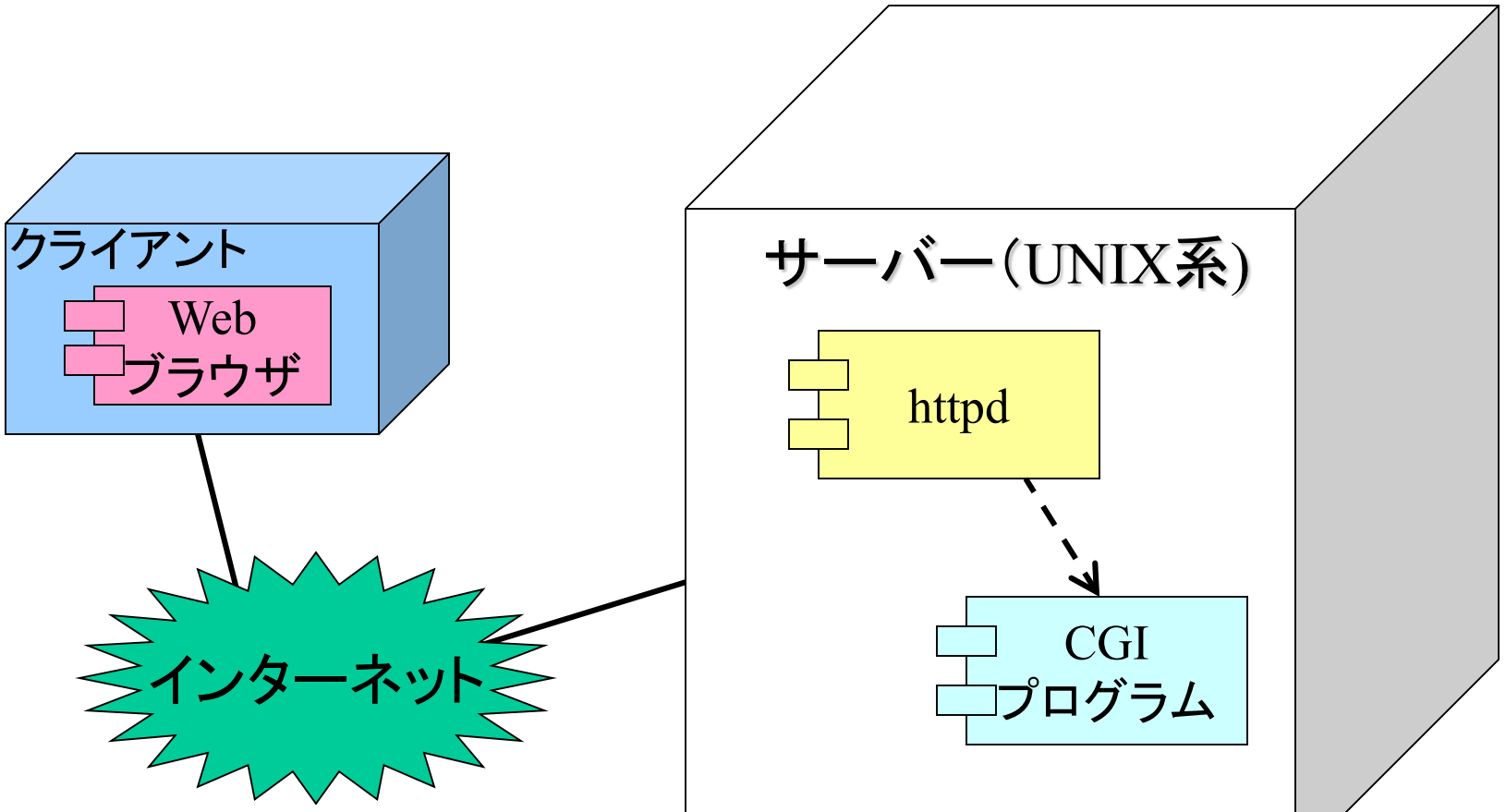
- いわゆるウエブサーバー.
- ホームページのデータを管理し，ブラウザからのリクエストがあれば，ブラウザにページのデータを送信するサービスを行う.
- httpdの種類や設定にもよるが，CGI等は，このhttpdの下請けとして他の(php等の)プロセスが走る.

httpdとクライアントの通信

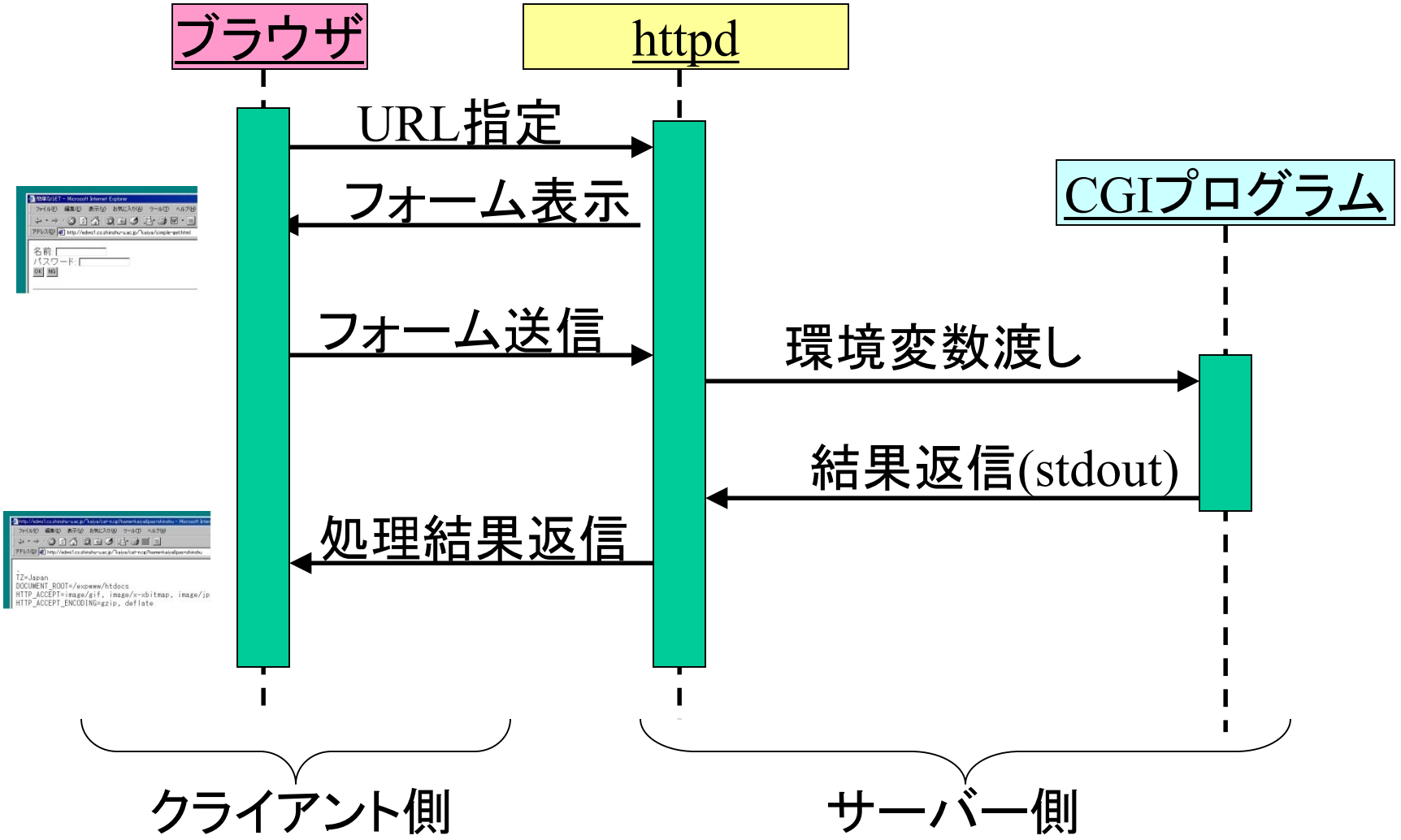
- 「*ページをクリックする*」に対応する内部的な命令
- ブラウザからサーバーに対して，見せて欲しいページを注文すること．



システム構成 (UML風に)



CGI実行のシーケンス例

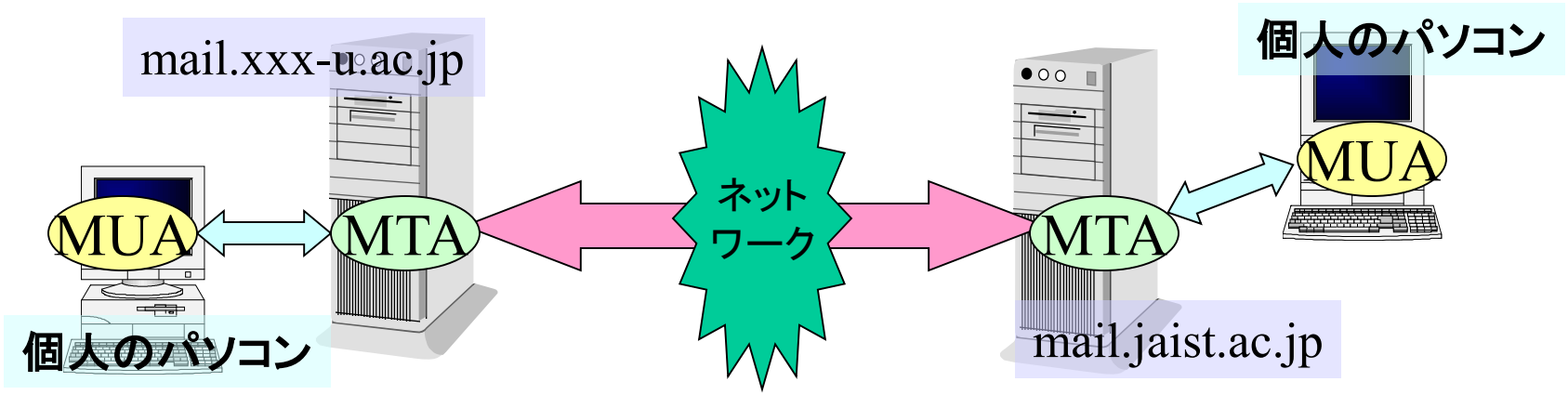


sendmail, postfix

- 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.
- 自前のMTAを使わず，組織全体のサーバーのMTAに接続してメール送信する場合がほとんどになった.
- 受信はまた別.

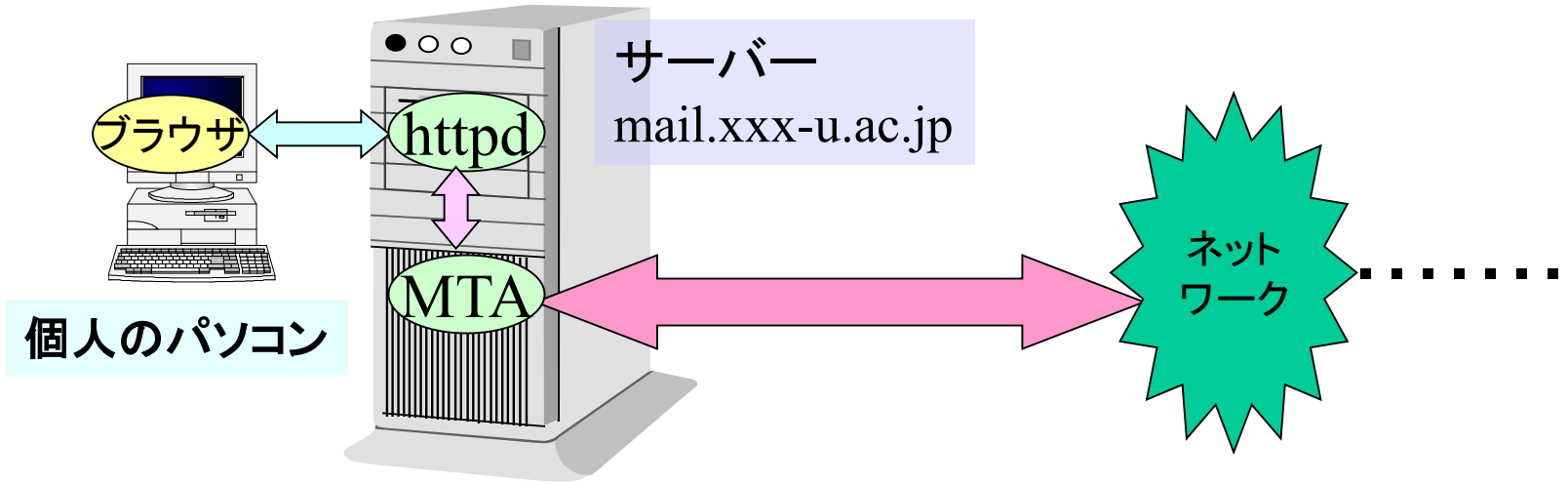
補足

- MUA: メールユーザーエージェント
  Outlook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デーモン
  sendmail, qmail など.



Webメールは？

- ご存じの通りブラウザ経由でメールが読み書きできる.
- コンピュータ中では，ウエブサーバー(httpd)とMTAが連携している.



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.

本日は以上

アンケートのほう，
よろしくご提出ください