LP49の利用説明書

H_2O

平成21年6月15日

目 次

第Ⅰ部

ＬＰ４９の基本

第1章　LP49の基本構想

1.1　LP49構造の概要

OSは、応用プログラム (あるいはユーザ) に対して”サービス”や”リソース”利用を提供する基盤である。OSは、処理性能とともに、あるいはそれ以上に、信頼性・頑強性・維持管理性が重要である。従って、構成要素のモジュール化、疎結合化（Ex. メッセージ結合)、障害の外波及防止と部分再開などが行い易いマイクロカーネル型OSが有利である。

ＯＳは多大な機能をもつだけに、スクラッチから全部を作るには巨大な資金とプロジェクトが必要である。幸いなことに、Ｌ４マイクロカーネル（独・カールスルーエ大学）と, Plan 9 OS (米・Bell研究所) という先進的な成果があり、かつ共にオープンソースとなっている。これを活用することにより、極小プロジェクト (約１名×数年) で我々の目標を実現しようというわけである。

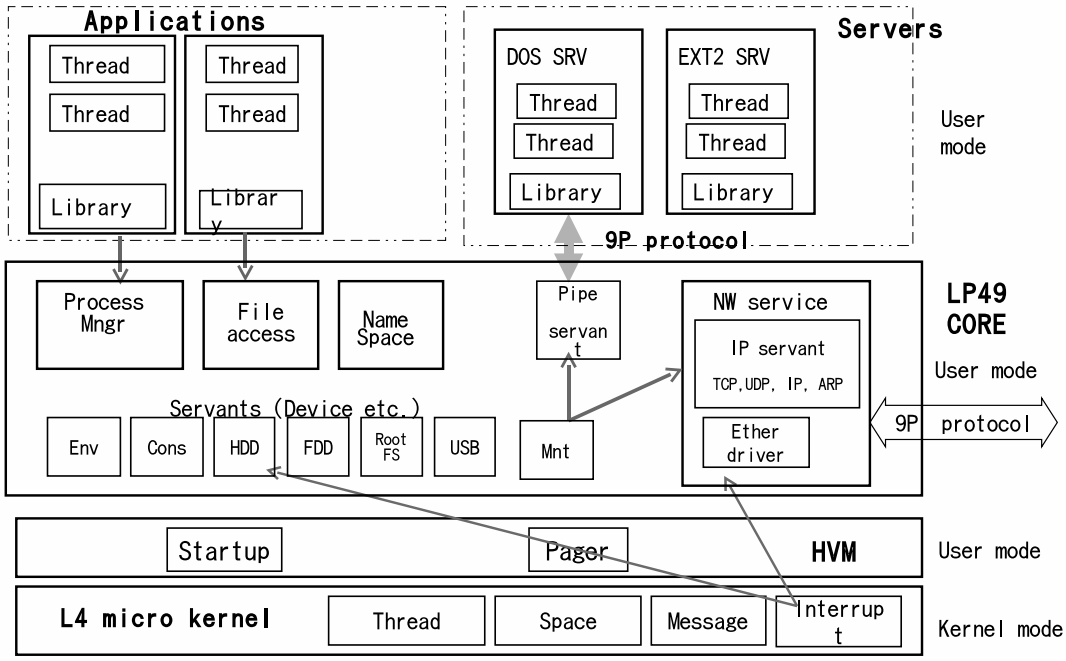


図 1.1: LP49 全体構成

LP49 は、図 1.1 に示すように、以下の階層かなできている。

L4 マイクロカーネル メモリ空間管理、スレッド管理、メッセージ通信を効率よく実現している。この部分だけがカーネルモードで動作する。

HVM (Hypervisor monitor) LP49 の startup と、メモリページ管理 (Pager) をおこなう。HVM という名前は、将来仮想マシンとしての機能をくみこむ意図で付けたが, 現在は仮想マシン機能は実装していない。ページ管理は現在のところ必要最低限しか実装していないが、L4 マイクロカーネルの強力と融通性を発揮できる分野であり、今後大いに発展させる予定である。

LP49-CORE 名前空間の管理、サーバのマウント, プロセス管理システムコールの処理、ファイル系システムコール（該当サーバ/サーバントに実行させる）の機能を持つ。

サーバ サーバは、9P プロトコル (Plan9 のサーバインタフェース) メッセージによって、サービスを提供する。メッセージは、TCP 接続 もしくは PIPE (ノード内の場合) を介してやり取りされる。

ライブラリ 応用プログラム APL は、使い慣れたシステムコール呼び出しでもってサービスを受けられる。ライブラリは、これをメッセージになおして CORE プロセスに送る。

シェル (qsh, rc) シェルとしては、qsh と rc を容易している。“Qsh” は LP49 のデバッグ用に作ったシェルで、色々なシステムコールを組込みコマンドとして提供している。“rc” は Plan9 の強力なシェルである (移植進行中)。qsh の中から起動できる。

APL: 応用プログラム

1.2 LP49 の基本発想

(1) マイクロカーネル

OS の疎結合モジュール化、デバイスドライバを含むプログラムのユーザモード化、個別プロセス再開による耐障害強化、マルチスレッドプログラミングの容易化のために、マイクロカーネル型 OS とした。マイクロカーネルは、スレッドとメッセージの効率が高く、融通性も高くかつシンプルな L4 を利用した。OS 全体を管理するモジュールは、“L4 マイクロカーネル” の 1 タスク (論理空間+スレッド) としてユーザモードで実行させている。これを “LP49-CORE” と呼ぶ。

(2) すべてのリソース・サービスをファイルトリーとして扱う

Unix では、すべてのリソースが階層的ファイルトリー上に名前付けされており（名前空間）、ファイルとしてアクセス出きることを目指したが、この理念はネットワーク (NW) をはじめとして早期に破綻した。LP49 では Plan9 を踏襲して、NW も含めてすべてアクセス対象をトリーディレクトリー上の名前で識別でき、ファイルインタフェース (open(), read(), write(), 等) で操作されるオブジェクトとして扱っている。

(3) サービス部品：サーバとサーバント

OS が提供するサービスには、DOS ファイルサービス、EXT2 ファイルサービスといった高位サービスと、ハードウェア駆動、モジュール間通信、NW 接続、サービス登録簿といった低位サービスとがある。

前者は、個々に独立性が高く、規模も大きくなりがちなので、サービス毎にユーザモードで走るプロセスとして実現する。これを**サーバ**と呼ぶ。ローカルでもリモートでも同等に使えるように、サーバはメッセージインタフェースとした。ユーザモードプロセスなので、プログラム開発の容易化のみならず、障害時もそのサーバだけを停止・再開することで耐障害性も強化される。

後者は、共通部品的であり、より実行速度が重視されるので、独立したプロセスとはせず LP49-CORE 内のモジュールとして実装することとした。このモジュールを**サーバント**と呼んでいる。サーバントは、統一インタフェースを持つコンポーネントである。同一サービスをサーバで実装することも、サーバントで実装することも可能である。

(4) マルチサーバと 9P プロトコル

サーバのプロトコル（メッセージインタフェース）は、サーバが提供できる機能と性能を決定する。いくら強力であっても、独自プロトコルは世の中に普及させることは至難である。幸い Plan9 の **9p プロトコル**は、`attach()`, `walk()`,`open()`, `create()`, `read()`, `write()`, `clunk()`, `remove()`, `stat()`, `wstat()` 等のメッセージからなり、低レベル制御も可能で融通性が高いので、これを採用した。9P プロトコルを採用した副産物として、少ない修正で Plan9 のサーバを LP49 に移植することも可能になった。

(5) 名前空間とその接続

前述のように、サーバもサーバントも内部のアクセス対象はファイルトリー上のファイルとして抽象化されており、UNIX でいうファイルシステムである。つまり, それぞれが名前空間をなしている。

図 1.2 に示すように、ルートファイルシステム (**RootFS**: 実はこれもサーバント) を出発点とする**名前空間**に、サーバやサーバントをを接続（マウント）することにより、ユーザに見えるようにする。“/dev” には各種デバイスサーバントが、“/net” にはプロトコルスタックが接続されている。サーバはリモートの可能性もあるので、マウントの仕組みは後で説明する。

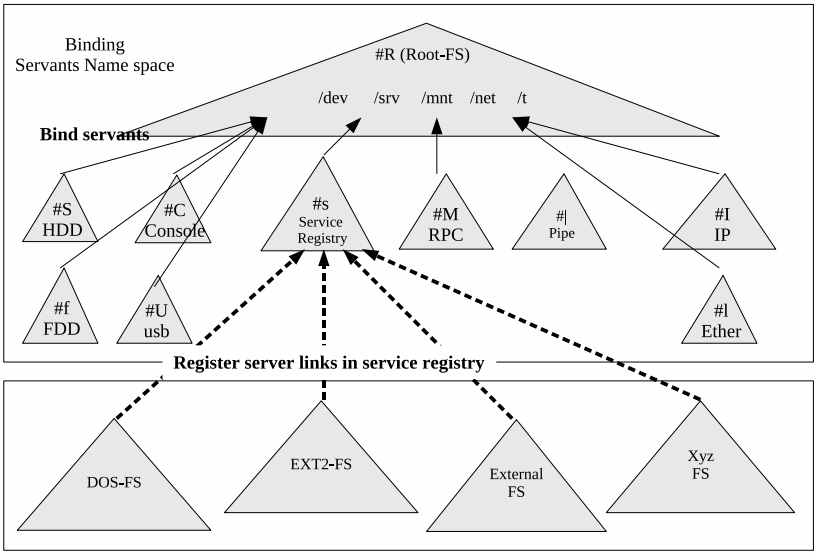


図 1.2: サーバ、サーバントと名前空間

(6) プロセス個別名前空間

UNIX ではファイルシステムの mount は root のみが行え、名前空間は全プロセスで共通である。これに対し、Plan9/LP49 の名前空間は、各プロセス毎に自前の名前空間（**個別名前空間**）を持つことができる。名前空間はアクセス保護の役目を持つので、緻密なセキュリティー管理を実現できる。

第2章　LP49の立ち上げ方

2.1　ブートメディア

LP49 は、ブートローダとして GNU GRUB を用いている。LP49 は、実機 (PC-AT), qemu エミュレータ、VMware 仮想マシンなどの上で走る。なお、以下の説明は作業環境として Linux を想定している。LP49 のブートは、現在の所 CD-ROM 及びフロッピーディスク (1.4MB オブジェクトと 2.8MB) からブート可能である。これらのイメージファイルは、LP49-yymmdd.tar ファイルに載せてある。フロッピーディスクは容量が小さすぎるので、今後は CD-ROM のみをサポートしていく予定である。

- lp49-boot.cd: CD-ROM イメージ (ISO9660 形式)
- lp49-boot.fd: 1.4MB フロッピーイメージ
- lp49-boot-x2.fd: 2.6MB フロッピーイメージ

(1) CD-ROM からのブートする

- 実機の場合
  CD-ROM イメージファイル lp49-boot.cd を CD-ROM に書き込み、CD-ROM からブートする。
- QEMU の場合
  CD-ROM イメージファイル lp49-boot.cd を用いて qemu を起動する。
  ```
  # qemu -cdrom lp49-boot.cd -std-vga
  ```
- VMware の場合
  CD-ROM イメージファイル lp49-boot.cd を用いて VMware を起動する。

(2) GRUB がメニューリストを表示する

LP49: CD BOOT (Hvm + Pc + Qsh + DosSrv + 9660Srv) デバグ用シェル “qsh” が使われる。

LP49: CD BOOT (Hvm + Pc + Init + DosSrv + 9660Srv) Plan9 の強力シェル‘ “rc” が使われる。

....

今のところ、最初のデバグ用シェル “qsh” を選択すること。2 番めの Plan9 の強力シェル‘ “rc” はまだ開発中であり、動かない部分がある。

(3) LP49 が立ち上がり、CD(default) か FD かと聞いてきたら Return キイを押す

以上により、LP49 のプロンプト (“qsh” の場合　`LP49[/]:` , “rc” の場合は　`[lp49]`　) が表示されるので、Unix 類似のシェル組み込みコマンド (`ls`, `cd`, `echo`, `cat`, `pwd` など) を入力して、動作を確かめられたい。

2.2　名前空間: ディレクトリー

“Directory tree” は、いわゆるファイルだけでなくほとんどのリソースやサービスも含めて、階層的な名前（パス名）でアクセスするための仕組みである。つまり、リソースやサービスを名前で識別するので ” 名前空間” とよぶ。

LP49 は、立ち上げ時に各種のリソースやサービスを名前空間につなぐ。立ち上げたときの名前空間を以下に示す。

【名前空間＝ディレクトリ構成】

```
--- / ---+-- bin/    <- - - ここには各種コマンドが見える
         |                 (実際には /t/bin/ がマウントされている)
         |
         |
         |-- dev/ --+-cons        ← 各種デバイスがバインドされている。
         |          |   :
         |
         |-- env/       ← 環境変数
         |-- fd/
         |-- net/    <- - - ネットワーク関係が見える
         |
         |-- proc/   <- - - Proc ファイル
         |-- srv/    <- - - サーバ登録簿 (service registray)
         |-- root/
         |-- n/      <- - - リモートのサーバをマウントする場所
         |-- c/
         |-- ums/
         |-- exportfs/
         |
         |-- t/ --+-bin/--+- コマンド          --*
         |        |       |   :                  | これらはブート CD に載っている。
         |        |                              |
         |        +-boot/---+- hvm.gz            | ブート CD のファイルシステムは、"/t"
         |        |         |- pc.gz             | にマウントされているので,
         |        |         |- qsh.gz            | この場所に見える。
         |        |                              |
         |        +-l4/  ---+-- kickstart.gz     |
         |                  |-- l4ka.gz          |
         |                  |-- sigma0.gz      --*
         |
         |-- rc/--- rcmain                     --* これらは、ルートファイルシステム
         |-- lib/ --- namespace                  | devrootfs.c が提供しており、/に
         |-- mnt/ --- exportfs/ --+- 0/ ---      | ユニオンマウントされているので、
         |                        |- 1/ ---      | この場所に見える。
         |                        |- 2/ ---      |
         |-- tmp/                                |
         |-- work/ --                            | RAM ファイルシステムなので、
         |-- usr/ --                             | ファイルの生成・書き込みも可能。
         +-- home/ ---+-- eve/                   |
                      |-- adm/                   |
                                               --*
```

UNIX の名前空間は、全プロセスに共通であるのに対し、LP49 の名前空間は各プロセス毎に自前の ”個別名前空間” を持つことができる（後述)。

2.3 コマンドを試してみる

(1) qsh シェル組み込みコマンドを入力してみる

```
LP49[/]: echo  abcdefghijklmn
LP49[/]: cd /t/bin
LP49[/t/bin]: ls
```

```
LP49[/t/bin]: cat ex2
```

(2) 実行ファイルを走らせてみる

“spawn <ファイルのパス名> <引数>”, もしくは spawn を省略して “<ファイルのパス名> <引数>” と入力する。同名のシェル組み込みコマンドがある場合には “spawn” を付ければ実行ファイルが実行される。

```
LP49[/t/bin]: spawn ls -l      <- - ls ファイルが実行される
LP49[/t/bin]: ls               <- - 組み込み ls コマンドが実行される
LP49[/t/bin]: ex2 2            <- - デモプログラムファイルが実行される
LP49[/t/bin]: cd /
LP49[/]: /t/bin/ex2 4
```

(3) Qemu の場合、フラクタル図を描かせてみる Qemu 上で走らせている場合は, 次のように入力してみよう。Mandelbrot のフラクタル図が表示される。

```
LP49[/]: d vga
```

(4) ファイルを作って書き込んでみる

LP49 の立ち上げ時に、“/tmp” には RAM ファイルシステムがマウントされている (この表現は厳密には正しくないことが、後で説明される)。したがって、このディレクトリーを使えばハードディスクを使わずともファイルを生成することが可能である。“qsh” はデバッグ用のシェルなので、create(), write(), read() などのシステムコールに対応したコマンドも組み込まれている。

【利用例】

```
*-----------------------------------
|    LP49[/]: cd  /tmp
|    LP49[/tmp]: create  -w  zzz 0666
|       ....  FD の値を返す .....   ここでは FD=10 とする。
|    LP49[/tmp]: write  10   AAABBBCCCDDDEEEFFFGGGHHH
|    LP49[/tmp]: pread  10  0
|       AAABBBCCCDDDEEEFFFGGGHHH
*-----------------------------------
```

(4) cmd.zip ファイルの展開

LP49 はファイル圧縮 zip, unzip も備えている。いくつかの代表的コマンドは ZIP ファイルに圧縮して t/bin/cmd.zip として載せてある。以下の手順で、このファイルを /tmp に展開することができる。

【利用例】

```
LP49[/]: cd  /usr
LP49[/tmp]: unzip  -f  /t/bin/cmd.zip  <- - ブート CD 上の t/bin/cmd.zip を解凍する。
LP4p[/tmp]: ls
```

【参考】ブート CD-ROM の内容

TAR ファイル LP49-yymmdd.tar には、CD-ROM イメージ lp49-boot.cd が用意されている。ブート CD-ROM イメージには、以下が含まれている。

```
---+--boot/ --+--grub/ --+-menu.lst
   |          |          +-stage2-eltorito (grub のプートコード)
   |          |
```

```
|          |--hvm.gz          ← LP49 起動とページャー
|          |--pc.gz           ← LP49 Core プロセス
|          |--qsh.gz          ← LP49 簡易シェル
|          |--dossrv.gz       ← LP49 DOS ファイルサーバー
|
+--l4 ----+ -kickstart-0707.gz        ← L4 の起動
|         | -l4ka-0707.gz             ← L4 マイクロカーネル
|         | -sigma0-0707.gz           ← L4 の Pager
|
+--bin/ - -+--cmd.zip         ← いくつかのコマンドを入れた ZIP ファイル
           |--...
           |--srv             ← LP49 srv コマンド
           |--ex2             ← LP49  デモプログラム
           |--...
```

第3章　サーバント

3.1　サーバントとは

サーバントはハードウェアデバイス, サービスの登録簿 (後述)、環境変数記憶、プロトコルスタックなど低位サービスを提供する。Plan9 のカーネルデバイスファイルシステム (Kernel device file systems) に相当する。サーバントは LP49-CORE プロセス内のモジュールであり、attach(), init(), open(), read(), write(),,, といったプロシージャインタフェースで呼ばれる。

システムは、ルート (”/”) から始まる名前空間（Root file server）を持っている。各サーバやサーブレットも個別に名前空間 (directry tree) を持っており、各要素はパス名で指定される。サーバやサーブレットの名前空間を、自分の名前空間に”マウント” することにより、サービスにアクセスできるようになる。

3.2　サーバントの種類

サーバント (内部サーバ) の一覧を表 3.1 が示す。

表 3.1: サーバント

サーバント名	機能	補足説明
#c	コンソール	/dev/cons
#e	環境変数	/env
#l	Ether ネットデバイス	
#f	フロッピーディスク	
#I	IP プロトコル	/net
#p	proc ファイル	/proc
#X	ループバックリンク	
#M	Remote Procedure Call	/mnt, サーバをマウントする機構
#|	パイプ	
#R	Root ファイルシステム	/ (Plan9 とは異なる)
#S	ストレッジデバイス	/dev/sdC0, /dev/sdD0
#s	サーバ登録簿 (Server registry)	/srv
#U	USB ホストコントローラ	
#v	VGA コントローラ	

3.3　サーバントの接続（バインド）

サーバントは、”bind コマンド” により、名前空間の指定マウントポイントに接続される。Unix では同一のマウントポイントに複数を接続した場合、以前のものは隠されて最後の接続のみが有効になるが、Plan9/LP49 では同一のマウントポイントに複数を接続することができる。これを **ユニオンマウント** と呼ぶ。ユニオンマウントの場合、同一の名前があれば前にあるものが見えるようになる。前に接続するか、後ろに接続するかは、bind/mount コマンドのオプションで指定する。

-a “after” を意味し、既存のものの後ろに接続される。

-b “before” を意味し、既存のものの前に接続される。

（無指定） 既存のものを置き換える。

また、 “-c ” オプションにより、Read-only で接続するか writable で接続するを指定できる。“-C ” オプションにより、Cache をするか否かを指定できる（まだ未実装)。

-c 指定すれば writable になる。

-C 指定すれば cache が有効化される（未実装)。

```
*-- << bind command >> -------------------------------------------
|  bind  [オプション]  サーバント名  マウントポイント
|
|  オプション   (無)  置き換え
|               -b    Before  (Cf. UNION mount)
|               -a    After
|               -c    書き込み許可
|
|-- 【例】-----------------------------------------------------
| bind -a #f /dev
*-------------------------------------------------------------------
```

LP49 のデバッグシェル qsh は、初期設定時に下記のコマンドを発行して各サーバントを所定位置に接続しているので、始めから各サーバントは名前空間に現れている。

```
bind  -a #c  /dev          ← コンソール
bind     #d  /fd           ← dup ファイル記述し記録
bind  -c #e  /env          ← 環境変数記録簿
bind  -c #s  /srv          ← サービス記録簿
bind  -ac #R  /            ← Root ファイルシステム（後述）
bind  -a  #f  /dev         ← フロッピー
bind  -a  #I0  /net        ← プロトコルスタック
bind  -a  #l   /net        ← Ether デバイス
```

ここで、”/” に接続された #R は RAM を使った Root ファイルシステム (Cf. src/9/port/devrootfs.c) である。これを使えば、ハードディスクを用いずとも、/tmp の下でファイルを create, write, read できる。

第4章　サーバ

4.1　サーバとサーバリンク

サーバは、9P プロトコルを喋れるユーザモードのプロセスである。9P プロトコルは、TCP 接続もしくは pipe を通して運ばれる。LP49 の pipe は両方向通信である。勿論 pipe は、同一ノード内でしか使えない。サーバにつながれた TCP 接続と pipe を合わせて「サーバリンク」と呼ぶことにする。サーバリンクにアクセスして 9P プロトコルをやり取りすることで、サーバからサービスを受ける。サーバリンクを登録しておく仕組みが「サーバ登録簿」である。LP49 は分散 OS なので、サーバ登録簿にはローカルサーバもリモートサーバも登録される。

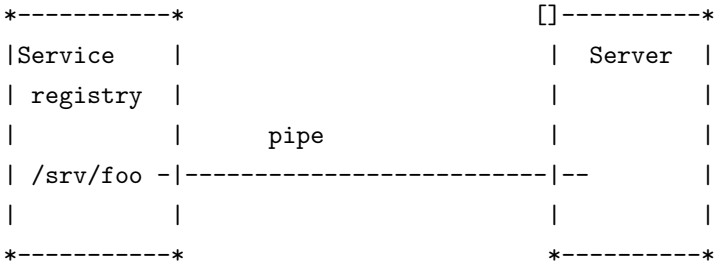
サーバ登録簿は "#s" という名前のサーバントであり (ソースプログラムは src/9/port/devsrv.c)、初期設定により名前空間の "/srv" に接続されている。サーバが登録されると "/srv/dos" (DOS ファイルサーバ)、"/srv/ext2" (EXT2 ファイルサーバ) のように名前空間に現れる。

4.2　サーバ登録簿 (/srv)

サーバは、起動されると以下のような手法で、サーバリンクを生成・登録する。

(1) Pipe 方式

サーバが pipe サーバリンクを生成を生成して、その file descriptor を自ノードの "/srv" に登録する。



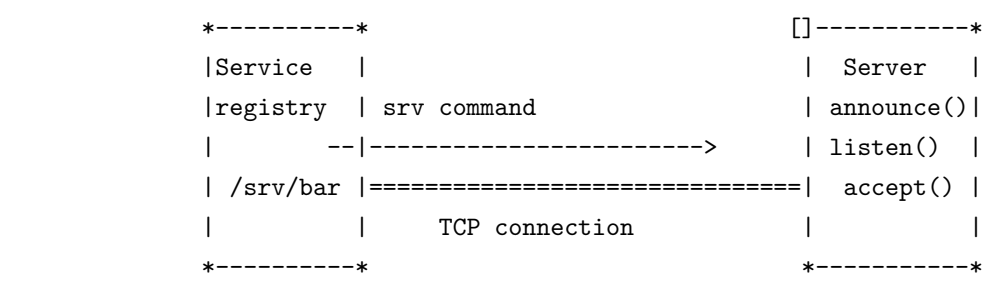
(2) Announce, listen, accept 方式と srv コマンド

サーバ側では、TCP のサービスポートを announce( ) して、サービス接続要求が来るのを listen( ) している。クライアントからの接続要求に対し、受け入れ可能だったら accept() してサービスを提供する。

クライアントマシン側では、"srv" コマンド (プート CD の /t/bin/srv) を用いて、サーバに接続要求を送る。サーバ接続が成功すると、サーバリンク (TCP 接続) が生成され、その file descriptor がサーバ登録簿に登録される。

```
*- << srv command >> ------------------------------------------------
|  /t/bin/srv [-abcC q]    tcp!サーバマシン!サービス     サービス名
|
|--【例】-------------------------------------------------------------
```

```
|    /t/bin/srv  -a tcp!10.0.0.1  u9fs
*-------------------------------------------------------------------
```



4.3 名前空間へのマウント

「サーバ登録簿」のサービスを名前空間にマウントすることにより、ローカル・リモートを問わずに同等にアクセスできるようになる。

```
*-- << mount command >> ------------------------------------------------
|      mount [オプション...]   パス名   マウントポイント   [位置指定]
|
|○オプション
|        なし:  置き換える
|        -a:    after
|        -b:    before
|        -c:    create 可能
|        -C:    Cache 使用
|
|○パス名
|     ex.  /srv/ramfs
|
|○マウントポイント
|     ex.  /work
|
|○位置指定
|      ファイルシステムのどの directory に接続するかを指定。
|     ex.  /dev/sdC0/ext2-0
|
|--【機能】-----------------------------------------------------------
| 「パス名」で指定された対象が、現名前空間の「マウントポイント」にマウントされる。
|
|--【例】-------------------------------------------------------------
|  mount  -a  /srv/u9fs   /n
|  mount -ac  /srv/ext2  /work  ""
|  mount  -ac  /srv/ext2  /work   /dev/sdC0/ext2-0
|
*-------------------------------------------------------------------
```

第II部

ストレッジサービス

第5章　RAMファイルサーバの使い方

5.1　RAMファイルサーバの使い方 (その1)

RAM ファイルサーバは、引数無で起動すると 直接 “/tmp” にマウントされる。

```
*-------------------------------------------------------------------
| 1. RAM ファイルサーバ RAMFS ( /t/bin/ramfs ) を起動 (spawn) する
|    LP49[/]: spawn /t/bin/ramfs
|              ←  RAMFS プロセスが生成され、/tmp にマウントされる。
|                  引数 -D を付けると、9P プロトコルのやりとりも出力される。
|
| 2. ファイルをつくってみる
|    LP49[/]: cd  /tmp
|    LP49[/tmp]: create -w  zzz  0666
|              ←  FD が返る。ここでは 10 と仮定する。
|    LP49[/tmp]: write  10  12345678901234567890
|    LP49[/tmp]: pread  10  0
|      12345678901234567890
|    LP49[/tmp]: close  10
|    LP49[/tmp]: mkdir  bb
|    LP49[/tmp]: cp  zzz  bb
*-------------------------------------------------------------------
```

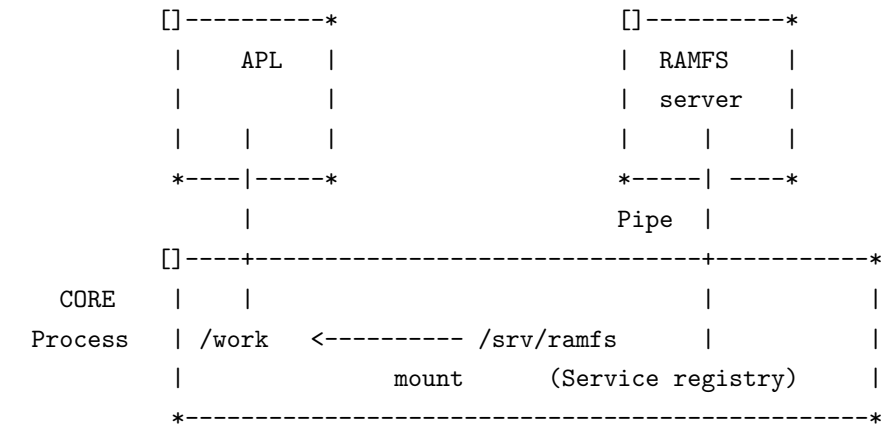
5.2　RAMファイルサーバの使い方 (その2 /srv)

RAM ファイルサーバは、引数 ”-S サービス名” を付けて起動すると、サービス記述簿 に “/srv/サービス名” として登録される。これを名前空間の適当な場所にマウントすることができる。

```
*-------------------------------------------------------------------
| 1. RAM ファイルサーバ RAMFS ( /t/bin/ramfs ) を起動 (spawn) する
|    LP49[/]: spawn /t/bin/ramfs  -S ramfs
|              ←  RAMFS プロセスが生成され、/srv/ramfs が登録される.
|
| 2. RAM ファイルサーバ RAMFS を /work にマウントする
|    LP49[/]: mount -ac /srv/ramfs /work
|              ←   RAM ファイルサーバ が /work に表れる。
|
| 3. ファイルをつくってみる
|    LP49[/]: cd  /work
|    LP49[/work]: create -w  zzz  0666
```

```
|              ←  FD が返る。ここでは 10 と仮定する。
|    LP49[/]: write  10  12345678901234567890
|    LP49[/]: pread  10  0
|      12345678901234567890
*-----------------------------------------------------------------
```

【図解】



第6章　ハードディスクの利用

6.1　ハードディスクドライバ

ATA ハードディスクは、“#S” という名前を持ち、“/dev” に接続して使用する。
【手順】

```
*-------------------------------------------------------------------------
|  ◎ HDD は、次のコマンドで名前空間/dev に接続する。
|     LP49[/]  bind  -a  #S  /dev
|     ←     #S はハードディスクドライバを意味する
|
|  ◎ ハードディスクは  /dev/sdC0  (あるいは  sdD0 ?) に現れる
|     ls  /dev/sdC0  を打ち込むと、例えば次のように表示される。
|     LP49[/] ls  /dev/sdC0
|     d r-xr-xr-x          0    sdC0/    ......
|     this directory contains ...
|     - rw-r-----          0     ctl   .......    ← HDD の制御ファイル
|     x rw-------          0     raw   .......    ← HDD の生ファイル
|     - rw-r-----   ........     data  .......    ← HDD 全体
|     - rw-r-----   .......      dos-0 .......    ← HDD のパーティション (DOS タイプ)
|     - rw-r-----   .......      ext2-0 .......   ← HDD のパーティション (ext2 タイプ)
*-------------------------------------------------------------------------
```

6.2　PC 機のパーティションと LP49 のパーティション

(1) パーティション設定済みのハードディスクの場合

PC 機では、ハードティスクの先頭 512 バイトに MBR (Master Boot Recorder) が載っており、MBR の後部にパーティションテーブルが載っている。

(現時点) PC 機のバーティションのうち、基本パーティションでタイプが dos, ext2,plan9 のものだけが、LP49 にも見える。
● dos タイプのパーティションは、LP49 では /dev/sdC0 に dos-0, dos-1,,, という名前で見える
● ext2 タイプのパーティションは、LP49 では /dev/sdC0 に ext2-0, ext2-1,,, という名前で見える

(2) パーティション無設定のハードディスクの場合

LP49 は、ハードディスクにパーティションが設定されていないと、src/9/pc/pcf-config-l4.c の

```
char __bootargs[2048] =
   ......
```

```
"sdC0part=dos 63 1000/plan9 1000 2000￥n"
..... ;
```

の内容に従って、LP49 のパーティションを設定する。ただし、ファイルシステムを構成しないと、DOS や EXT2 ファイルサーバでは使うことができない。

6.3 パーティションとファイルシステムの設定

(1) 実機の場合

(方法 1) Partition Magic などのプログラムで設定する。

(方法 2) Linux をイストールすると、パーティションがつくられ EXT2 ファイルシステムが設定れさる。

(方法 3) Windows をイストールすると、パーティションがつくられ DOS ファイルシステムが設定れさる。

(2) VMware の上で走らせる場合

(方法 1) Partition Magic などのプログラムで設定する。

(方法 2) Linux をイストールすると、パーティションがつくられ EXT2 ファイルシステムが設定れさる。但し、インストールのときに Linux ではなく『その他』としておかないと、誤動作する?

(方法 3) Windows をイストールすると、パーティションがつくられ DOS ファイルシステムが設定れさる。但し、インストールのときに Windows ではなく『その他』としておかないと、誤動作する?

第7章　EXT2ファイルサーバーの利用

【起動法】

```
*-------------------------------------------------
| 1. EXT2 パーティションを持つハードディスクを接続する。(既に接続されていれば、本コマンドは不要)
|    LP49[/]: bind  -a  #S  /dev
|    ==> ハードディスが /dev/sdC0 (あるいは sdD0 ?) に見えるようになる
|
| 2. EXT2 ファイルサーバを起動する
|    LP49[/]: spawn  /t/bin/ext2srv
|      ==> EXT2 ファイルサーバが起動され、サービス記録簿 /srv に /srv/ext2 が登録される。
|    オプション      -D :  9P プロトコルの内容を印字。 9P プロトコルのやり取りが見えて面白い。
|                    -v :  デバッグ情報を印字
|                    -r :  Read Only
|
| 3. EXT2 ファイルサーバ (/srv/ext2) を例えば名前空間の /work にマウントする。
|    LP49[/]: mount  -c  /srv/ext2  /work  /dev/sdC0/ext2-0
|    ==>ハードディスクの EXT2 パーティション  /dev/sdC0/ext2-0 が、/work にマウントされる。
|
| 4. EXT2 ファイルシステムを使う。
|    たとえば、Linux がインストールされていれば、-------
|    LP49[/]: cd  /work/etc
|    LP49[/work/etc]: cat  hosts
|      127.0.0.0      localhost.localdomain  localhost
|    LP49[/work/etc]: cd /work/bin
|    LP49[/work/bin]: open  -r  sort          ← Linux の /bin/ls ファイルを Open
|         FD が返される。ここでは 10 と仮定する。
|    LP49[/work/bin]: preadx  10  0  256      ← 先頭番地から 128 バイトをダンプする。
|         バイナリ情報がダンプされる。
|
*-------------------------------------------------
```

第8章　DOS ファイルサーバーの利用

【起動法】

```
*-----------------------------------------------
| 1. ハードディスクを接続する。(既に接続されていれば、本コマンドは不要)
|    LP49[/]: bind  -a  #S  /dev
|    ==> ハードディスクが /dev/sdC0 （あるいは sdD0 ?）に見えるようになる
|
| 2. DOS ファイルサーバは、既に立ち上がっており、 /srv/dos が見えるはず。
|    LP49[/]: ls   /srv
|       d  r-xr-xr-x     0     srv/   ..........
|       this directory contains ....
|       - rw-rw-rw       0     dos    .....
|
| 3. DOS ファイルサーバ（/srv/dos）を例えば名前空間の /work にマウントする。
|    LP49[/]: mount  -cc  /srv/dos  /work  /dev/sdC0/dos-0
|    ==>ハードディスクの DOS パーティション  /dev/sdC0/dos-0 が、/work にマウントされる。
|
| 4. DOS ファイルシステムが/work の下に見えるようになる。     ←←    バグがあり。
|    LP49[/]: ls /work
|
| 5. DOS ファイルを書いてみる
|    LP49[/]: cd  /work
|    LP49[/work]: create  -w  z1 0666
|       .... エラーメッセージを吐きながら FD の値を返す .....   ここでは FD=10 とする。
|    LP49[/work]: write  10   AAABBBCCCDDDEEEFFFGGGHHH
|    LP49[/work]: pread  10  0
|       AAABBBCCCDDDEEEFFFGGGHHH
|
*-----------------------------------------------
```

第9章　USBメモリの利用

USB は大変便利であるが、USB デバイスを使うためのプログラムは、簡単ではない。USB 規格には、USB-1.0 と USB-2.0 の規格がある。また、ホストコントローラインタフェースとして OHCI と UHCI がある。このうち、、現 LP49 がサポートしているのは USB-1.0 の UHCI のみである。最近のパソコンは殆ど USB-2.0 がついているが、その場合でも USB-1.0 のハブを間にかませれば、使うことができる。

ソースプログラム: src/9/pc/devusb.c、src/9/pc/usbuhci、src/cmd/usb/usbd/* 、src/cmd/usb/usbsfs/*

【起動法】

```
*-------------------------------------------------------------------------
| 1. USB を接続する。(既に接続されていれば、本コマンドは不要)
|    LP49[/]: bind  -a  #U  /dev
|    ==> USB が /dev/usb0 （あるいは usb1 ?）に見えるようになる
|
| 2. USB デーモンを起動する。
|    LP49[/]: spawn  /t/bin/usbd
|    → "usbd"プロセスが生成される。
|
| 3. usbsfs プロセスを立ちあげる
|    LP49[/]: spawn  /t/bin/usbsfs
|    → "usbsfs"プロセスが生成され、"/ums" にマウントされる。
|        /ums/ctl, /ums/raw,  /ums/data などを確認できるはず。
|
| 4. DOS2 ファイルサーバを立ちあげる
|    LP49[/]: spawn  /t/bin/dos2  -f  /ums/data:32  usbs
|    → "dos2"サーバが生成され、サーバ登録簿に /srv/dos2 が登録される。
|
| 4. usbsfs をマウントする
|    LP49[/]: mount -ac  /srv/usbs   /work
|    LP49[/]: ls  /work
*-------------------------------------------------------------
```

第III部

ネットワークサービス

第10章　マシン環境の設定

LP49 は、実機、Qemu エミュレータ、 VMware 仮想マシンなどで走らせることができる。
(1) 現在の所、下記の Ether カードが利用可能である
* AMD79c970 (Lance)
* NE2000
* Realtek 8139, 8168
* DEC2114
* Intel82557

(2) LP49 は、自動的に probing して Ether カードを判別して適切なドライバを選択する

(3) IP アドレスと Gateway アドレスの設定は、ipset コマンドを使う

```
*-------------------------------------------------*
| LP49[/]: ipsetup <IPaddress>  <GatewayAddress>    |
*-------------------------------------------------*
```

第11章　QEMUゲストオンリーNW(LP49-LP49通信)の設定

(1) 位置づけ
ホスト OS 上では、ゲスト OS (LP49) の ネットワークは TUP インタフェースとして現れる。

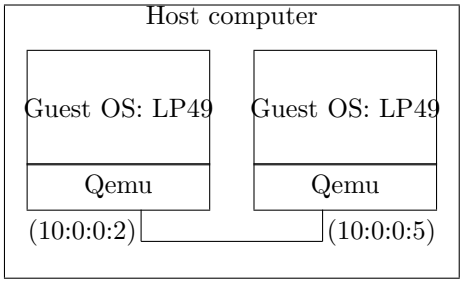


図 11.1: LP49 間通信

(2) LP49-1 の起動
LP49 ディレクトリにブート CD/FD イメージ (lp49-boot.cd, lp49-boot-x2.fd) があるものとする。
(2-1) QEMU の起動

```
+-------------------------------------------------------------------------
|  # cd  LP49
|  # qemu -cdrom  lp49-boot.cd  -std-vga -net nic -net socket,listen=localhost:1234
|       ←  CD イメージから立ち上げる場合
|           フロッピーイメージから立ち上げる場合は、『-fda lp49-boot-x2.fd』を指定する。
+-------------------------------------------------------------------------
```

(2-2) LP49 が立ち上がったら、IP アドレスを設定する

```
*-------------------------------------------------------------------------
|  LP49[/]: ipsetup
|     引数を指定しないと、IP アドレス: 10.0.0.2、 Gateway アドレス: 10.0.0.1 に設定される。
*-------------------------------------------------------------------------
```

(3) LP49-2 の起動
(3-1) QEMU の起動

```
*-----------------------------------------------------------------------------
|  # cd  LP49
```

```
|  # qemu -cdrom lp49-boot.cd  -std-vga -net nic,macadds=52:54:00:12:34:57 ¥
|         -net socket,connect=localhost:1234
|       ←  CD イメージから立ち上げる場合
|           フロッピーイメージから立ち上げる場合は、『-fda lp49-boot-x2.fd』を指定する。
*-------------------------------------------------------------------------------------
```

(3-2) LP49 が立ち上がったら、IP アドレスを設定する

```
*-------------------------------------------------------------------
|  LP49[/]: ipsetup 10.0.0.5
|    Gateway アドレス を省略すると 10.0.0.1 に設定される
*-------------------------------------------------------------------
```

(4) ping の実験

例えば LP49-1 から LP49-2 に ping を行なう。

```
*---------------------------------------------*
|  LP49[/]: /t/bin/ping  10.0.0.5             |
*---------------------------------------------*
```

(5) exportfs, import の実験

例えば LP49-1 の名前空間を LP49-2 にマウントする。

```
*-- <<LP49-1>> ----------------------------------------*
|  LP49[/]: /t/bin/listen1 tcp!*!55 /t/bin/exportfs &   |
*-------------------------------------------------------*
```

これにより、ノード LP49-1 は、任意のノードから TCP のポート 55 番 (tcp!*!55) への要求を監視し、この要求がきたらプロセスを生成して /t/bin/exportfs (名前環境を export するサーバ) を実行させる。

```
*-- << LP49-2 >> ----------------------------------------
|  LP49[/]: /t/bin/import -a 10.0.0.2   /t  /n
|  LP49[/]: cd   /n       ← /n に相手マシンの /t 以下の名前空間が見えるようになる。
*-------------------------------------------------------
```

これにより、ノード LP49-2 は、IP アドレス= 10.0.0.2 の名前空間 /t を自分の名前空間の /n にマウントする。”-a” はユニオンマウントの after を意味する。/n 以下を覗けば、相手マシンの /t 以下をみることができる。

第12章　TUN/TAPを使ったQEMU(LP49-ホストOS通信)の設定

(1) 位置づけ
ホスト OS 上では、ゲスト OS (LP49) の ネットワークは TUP インタフェースとして現れる。
(2) ホスト OS 側の設定
(2-1) TUN モジュールをインストール

```
*-----------------------------------------------*
|  # ls  -l  /dev/net/tun                       |
|  # chmod  0660  /dev/net/tun                  |
|  # chgrp   maruyama  /dev/net/tun             |
|  # modprobe  tun                              |
*-----------------------------------------------*
```

(2-2) "/etc/qemu-ifup" スクリプトを設定する

```
*--------- /etc/qemu-ifup ------------------------*
| #!/bin/sh                                       |
| sudo  /sbin/ifconfig  "\$1"  10.0.0.1           |
*-------------------------------------------------*
```

(2-3) "qemu-ifup" を実行できるよう、"visudo"コマンドを用いて "/etc/sudoers" ファイルに以下を追加

```
*------/etc/sudoers---------------------------*
| maruyama    ALL=(ALL)  ALL                   |
*-----------------------------------------------*
```

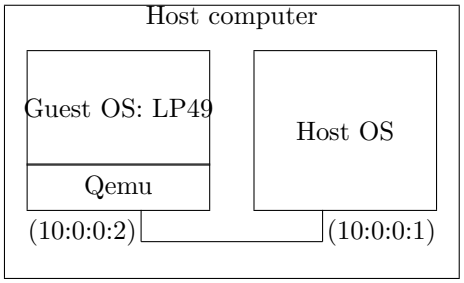


図 12.1: LP49-Linux 間通信

(3) Qemu コマンドにより、LP49 を走らせる

```
*-------------------------------------------------------------------------------*
| # sudo  qemu  -fda  /home/maruyama/LP49/lp49-bootfd-x2.fd  -net nic  -net tap  |
| password:   <your-password>                                                    |
*-------------------------------------------------------------------------------*
```

(4) LP49 上で、"ipsetup" コマンドにより IP アドレスと Gateway アドレスを設定する

IP アドレスと Gateway アドレスを省略すると、(qemu に便利なように) 10.0.0.2 と 10.0.0.1 が設定される。

```
*-----------------------------------------------------------*
| LP49[/] ipsetup <IPaddress>  <GatewayAddress>             |
+-- Ex.-----------------------------------------------------*
| LP49[/] ipsetup  10.0.0.2  10.0.0.1                       |
|    LP49[/] ipsetup                                        |
*-----------------------------------------------------------*
```

第13章　VMwareの場合の設定

(1) VMware のプライベットアドレスを調べる

```
*-------------------------------------------------------------------------------*
|  [Linux] /sbin/ifconfig                                                       |
|  lo        Link encap:Local Loopback                                          |
|            inet addr:127.0.0.1  Mask:255.0.0.0                                |
|            ............                                                       |
|                                                                               |
|  vmnet1    Link encap:Ethernet  HWaddr 00:50:56:C0:00:01                      |
|            inet addr:192.168.74.1  Bcast:192.168.74.255  Mask:255.255.255.0   |
|            ......................                                             |
|                                                                               |
|  ==>  Private NW       192.168.74.0/24                                        |
|       Gateway address  192.168.74.1                                           |
*-------------------------------------------------------------------------------*
```

(2) IP アドレスと Gateway アドレスを設定する

```
*-----------------------------------------------------------------------*
| LP49[/]: ipsetup  <IPaddress>  <GatewayAddress>                       |
+--Ex. ---------------------------------------------------------------+
| LP49[/]: ipsetup  192.168.74.2  192.168.74.1                          |
*-----------------------------------------------------------------------*
```

第14章　サーバ登録簿

LP49 では、サーバはサーバ登録簿 (/src/*) に登録されて、利用可能になる。サーバ登録簿については、第 1 部 を参照のこと。

第15章　離れたマシンの名前空間をマウントする

別のマシンの名前空間の任意のサブトリーを自分の名前空間のマウントポイントにマウントすることにより、別マシンのサブトリーにアクセスすることが可能にくなる。Plan9/LP49 ではデバイスを含めて全てのリソースはファイルシステムとして抽象化されているので、離れたマシンのデバイスも制御できる。

指定された名前空間へのアクセスの可否は、認証サービスの役目である。LP49 では認証サービスは未実装である。

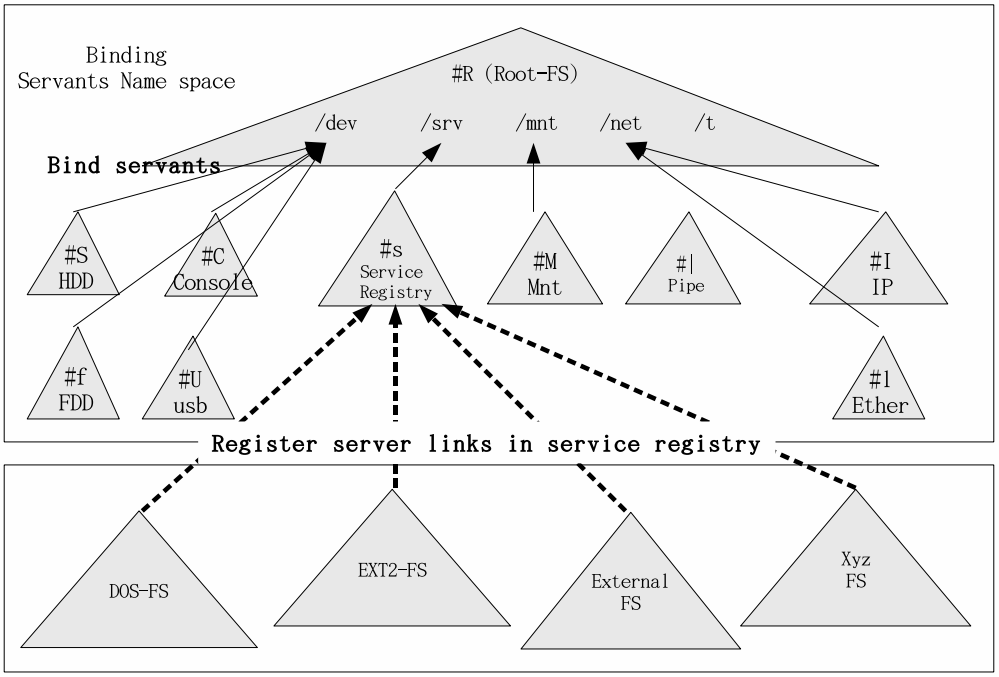


図 15.1: 名前空間とマウント

15.1　Export 側の設定

```
*-----------------------------------------------------------------
| 1. # cd LP49
|
| 2. # qemu -cdrom lp49-boot.cd -std-vga -net nic -net socket,listen=localhost:1234
```

```
|
| 3. LP49 を立ち上げる
|
| 4. LP49[/]: ipsetup
|    IP address is set to default value: 10.0.0.2
|
| 5. LP49[/]: t/bin/listen1 tcp!*!55 /t/bin/exportfs &
|    これにより、TCP 55 番ポートに要求がきたら、/t/bin/exportfs プロセスが起動される。
|
| 6. もし、9P プロトコルのやりとりを観察したいなら、-d を指定する。
|    LP49[/]: t/bin/listen1 tcp!*!55 /t/bin/exportfs -d &
|    -d を付けると、9P プロトコルのやりとりが印字される。
*-------------------------------------------------------------------
```

15.2 Import 側の設定

```
*-------------------------------------------------------------------------
| 1. # cd LP49
|
| 2. # qemu -cdrom lp49-boot.cd -std-vga -net nic,macaddr=52:54:00:12:34:57 ¥
|            -net socket,connect=localhost:1234
| 3. LP49 を立ち上げる
|
| 4. LP49[/]: ipsetup 10.0.0.5
|    IP address is set to the specified value: 10.0.0.5
|
| 5. LP49[/]: t/bin/import -a 10.0.0.2 /t /n
|    これは、マシン 10.0.0.5 のディレクトリー /t 以下の名前空間を
|    自マシンのィレクトリー /n にマウントすることを要求している。
|
| 6. LP49[/]: cd /n
|    /n に export マシンの名前空間 /t が見えるようになる。
|
| 7. LP49[/n]: ls
|
| 8. LP49[/n]: cd bin
|
| 9. LP49[/n/bin]: ex2
|    export マシンの /t/bin/ex2 が実行される。
|
| 10.LP49[/n/bin]: cd
|
| 11.LP49[/n/bin]: unmount  /n
*-------------------------------------------------------------------------
```

第16章　U9FS: Linuxのファイルトリーを LP49マウントする

U9FS は、Linux のファイルトリーを LP49 の名前空間にマウントして、Linux 上のファイルをアクセスする機能である。Plan9 の u9fs を LP49 への移植である。

16.1　Qemu・VMwawe 上の LP49 にホスト OS のファイルをマウントする例

16.2　Linux 側の設定

Linux 側では、”u9fs” プログラムが走るように設定する。

1. 実行ファイル ”u9fs” (LP49-yymmdd/src/cmd/unix/u9fs/u9fs に入っている) を”/usr/local/etc” にインストールする。

```
*-------------------------------------------------------------------------*
| [Linux]: cp LP49-yymmdd/src/cmd/unix/u9fs/u9fs /usr/local/etc       |
*-------------------------------------------------------------------------*
```

2. ”/etc/services” に下記行を追加する

```
*-------------------------------------------------------*
| u9fs      564/tcp                                     |
*-------------------------------------------------------*
```

3. 以下を内容とするファイル ”u9fs” を ”/etc/xinetd.d/” に設ける。

```
 *-------------------------------------------------------------*
 |   service u9fs                                             |
 |   {                                                         |
 |     disable         = no                                    |
 |     socket_type     = stream                                |
 |     protocol        = tcp                                   |
 |     user            = root                                  |
 |     wait            = no                                    |
 |     user            = root                                  |
 |     server          = /usr/local/etc/u9fs                   |
 |     server_args     = -Dz -a none -u maruyama /home/        |
```

```
|     groups          = yes                                    |
|     flags           = REUSE                                  |
|   }                                                          |
|                                                              |
*--------------------------------------------------------------*
```

ここに"server_args" は 以下を意味する。

-Dz : デバッグデータの置き場所　"/tmp/u9fs.log"

-a none : No authentification.

-u <username> ; /home/<username>

/home/ : File trees from /home/ are mounted.

4. Firewall の tcp/564 ボートを空ける。

   (FedoreCore の Firewall 設定には、結構苦心した)

5. xinetd をリスタートする

```
*-------------------------------------------*
|  % /etc/init.d/xinetd  restart            |
*-------------------------------------------*
```

16.3　LP49 側の設定

LP49 では、”srv” コマンドを使って、Linux 上の u9fs サーバへのサーバリンクをえる。”srv”コマンドは、ブートフロッピーに ” b/srv” として載せてある。

1. LP49 を起動する

2. IP アドレスと Gateway ヲドレスを設定する。

```
*-------------------------------------------------------------*
|  LP49[/]; ipsetup  <IPaddress>  <GateWayAddress>            |
|-【例】-------------------------------------------------------+
|  Ex.  ipsetup 10.0.0.2  10.0.0.1                            |
*-------------------------------------------------------------*
```

3. ”srv”コマンド (ブートフロッピーの b/srv) を実行する。

```
*-----------------------------------------------------------*
|  LP49[/]; spawn  /t/bin/srv  tcp!10.0.0.1  u9fs            |
*-----------------------------------------------------------*
```

これにより、目的サーバの u9fs に対して TCP コネクションが張られ、サーバ登録簿に ”/srv/u9fs” が登録される。

4. サービスを名前空間の ”/n”にマウントする。

```
*-------------------------------------------------------------*
| LP49[/]; mount -a /srv/u9fs  /n                             |
| LP49[/]: cd  /n                                             |
| LP49[/n]: ls                                                |
*-------------------------------------------------------------*
```

5. ”/n” の下ニ、Linux のファイルトリーが現れる.

6. 仕事が終了したら、unmount する.

```
*-------------------------------------------------------------*
| LP49[/n]; cd /                                              |
| LP49[/]: unmount  /n                                        |
*-------------------------------------------------------------*
```

第IV部

qshシェルとrcシェル

第17章　QSHシェル

17.1　QSHシェルとは

qsh シェルは、LP49 のデバッグのために作った (dirty な) 簡易シェルであり、多くのシステムコールをコマンドレベルで試すことができるようになっている。本格的シェルとしては、Plan9 の ”rc” シェルを移植中であり、これは qsh の中から起動できる。qsh は、以下のシーケンスで起動される。ブートフロッピー上のファイルを使うために、DOS ファイルサーバも起動している。

```
HVM process                              Cf. src/9/hvm
   -->  PC process  (core process)       Cf. src/9/{pc, port, ip}
        -->  QSH process                 Cf. src/9/qsh
        --> DOSSRV process          Cf. src/cmd/dossrv
```

qsh は、初期設定として以下の名前空間を作り、サーバント (拡張デバイス) などを接続する

```
*-------------------------------------------------------------------
|    /dev       デバイス (#c=console, #f=floppy,,,) を接続する
|    /env       環境変数 (#e)
|    /srv       サーバ登録簿 (#s)
|    /          RAM ファイルサーバント (#R) を接続
|    /net       Protocol stack(#I), EtherDriver(#l)
|    /t         ブートフロッピーイメージをマウント
*-------------------------------------------------------------------
```

ファイルサービスとしては、”/t” にブートフロピーが、”/” に RAM ファイルサーバントがマウントされており、ハードディスクを使わずともファイル機能を試すことができる。

(a) ”/t” にはブートフロピーをマウントされている

```
-- / -- t/ --+- boot/ - -+ grub/ --menu.lst
             |           |-- hvm,gz   : Hypervisor process (only name)
             |           |- pc.gz    : LP49 core process (user mode, off course)
             |           |- qsh.gz   : Quasi shell (Shell for debugging)
             |           |- dossrv.gz : DOS file server
             |
             |--l4/  --|-kickstart-0804.gz  :  L4 startup
             |         |-l4ka-0804.gz       :  L4 micro kernel
             |         |-sigma0-0805.gz     :  L4 sigma0 pager
             |
             |
             |- bin/ --|- cmd.zip  : zip files containing several commands
                       |- srv      : srv command
                       |- ex2      : program example
                       |- rc       : Shell ported from Plan9
                              :
                              :
```

(b) ”/” に ROOT ファイルサーバント#R (cf. src/9/port/devrootfs.c) がマウントされている

17.2 QSH の組込みコマンド (その 1: 通常コマンド)

○ help

○ unzip -f <zipfile>
  Ex. cd /usr
      unzip -f /t/bin/cmd.zip --- Now, you can see several copmmands on /tmp.

○ tar [cxt][vo][F][f] <tarfile> [<files>...]
  -- Not yet implemented

○ ls [<path>] show the list of <path> directory
○ lc [<path>] short verion of ls

○ cd [<path>] changes working directory to <path>
○ pwd show the current working directory

○ echo <any_string> echo back any-string

○ cat <path> displays contents of the file <path>

○ remove <name> Delete a file Not-Yet-Implemented

○ mkdir <dirname> make a new directory

○ fd2path <fd> <buf> <len> Not yet implemented

○ bind [-abcC] <#newpath> <mountpoint>
  -a after
  -b before
  -c allow creattion -ac, -bc
  Ex. bind -a #l /net
      bind -a #SsdC0 /dev

○ mount [-abcC] <srvname> <mountpoint> <spec>
  -a after
  -b before
  -c allow creattion -ac, -bc
  -C
  Ex. mount -c /srv/dos /tmp /dev/fd0ctl

○ unmount [<name>] <mntpoint>

○ putenv <name> <value>
○ getenv <name>

○ rfork
○ fork [<pathname>]
○ exec <pathname>
○ exits <msg>

○ spawn <pathname> <parameters>....
○ <pathname> <parameters>....
--- Spawn a new process

○ ipsetup [<IPaddress> <GatewayAddress>]
-- Setup the IPaddress and GatewayAddress.
-- Default is: 10.0.0.2 10.0.0.1

○ usbsetup
-- Setyup the USB memory

○ partition
-- Get the partition table contents

【お遊び】
○ d vga マンデルブロー図
○ SL
○ pi pi-value
○ credits Contributors

17.3 QSH の組込みコマンド (その 2: ファイルサービステスト)

Qsh はデバッグ用シェルなので、以下のコマンドも組み込んでいる。

○ create [-rwWxc] <name> <permission>

○ open [-rwWxtcde] <name> <mode>
-r Read
-w Write
-x Executable
....

○ close <fd>

○ seek {fd> <offset> <type>

○ write <fd> <any_string>

○ pwrite <fd> <offset> <any_string>

○ read <fd>

○ pread <fd> <offset>

○ preadx <fd> <offset> [<size>]
show the contents in Hex and ascii of <fd> file

○ pipe

○ dup <oldfd> <newfd>

○ put <txt> _write <text>

○ putsl _write $sl

○ get displays contents _written

○ give <txt> _write <text>

○ givesl _write $sl

○ take displays contents _written

○ maptest <size> Test L4 fpage mapping

○ pgmap <procnr>

○ pgmap <procnr> <adrs> <len>

17.4 QSH の組込みコマンド (その 3: デバッグダンプ)

○ kd Enter into the L4 kernel bebugger

○ dump <process-nr> <start-adrs> <size> Memory dump

○ qsh は、デバッグダンプ機能として "d" コマンドをもっている。

```
*--------------------------------*
|    d  <table> <parameter>     |
*--------------------------------*
```

○ d chan <Chan-adrs>

○ d ch <Chan-fid>

○ d namec <path-name> Dump the chan denoted by <path-name>

○ d dev <Dev-adrs>

○ d mount <mount-adrs>

○ d mhead <Mhead-adrs>

○ d mnt <Mnt-adrs>

○ d pgrp <Pgrp-adrs>

○ d fgrp <Fgrp-adrs>

○ d proc <Proc-adrs>

○ d up Dump the current Proc-table

○ d ns Dump the current name space (Mhead, Mount,,,,)

○ d nss dumpmount() in port/chan.c

○ d dirtab <Dirtab-adrs>

○ d walkqid <Walkqid-adrs>

第18章　RCシェル

rc シェルは、Plan9 rc シェルであり、大変強力な機能をもっている。現在 LP49 への移植を進めているところであり、部分的に動作する。rc シェルは、ブートフロッピーイメージ lp49-boot-x2.fd に bin/rc に載せてあるので、qsh から以下のようにして起動できる。

```
LP49:[/]: /t/bin/rc
```

path 変数は "(. /t/bin /bin)" に設定してあるので、プログラムサーチは現ディレクトリ, /t/bin, /bin の順でおこなう。

【メモ】"rc" はコマンドのシンタックス分析に yacc を使っているが、Unix の yacc が short int を返すところを、Plan9 の yacc は int を返す (Unicode 対応?)。この違いに気がつくのが遅れ、時間をロスした。

第Ⅴ部

Makeの方法

第19章　Makeの仕方

19.1　LP49 tar ファイル

”LP49-yymmdd.tar” ファイルは、http://research.nii.ac.jp/H2O/LP49 からダウンロードしてください。

【Tar ファイル内のディレクトリ構造】

```
*-------------------------------------------------------------------------
|  LP49-yymmdd/                                                          |
|              doc/                                                      |
|              include/                                                  |
|              lib/      --- symbolic links to libxxx.a files            |
|              rootfs/                                                   |
|                      bin/    --- LP49 command files                    |
|                              Makefile    -- build cmd.zip              |
|                              cmd.zip                                   |
|                              ....        -- command files              |
|                      boot/   --- LP49                                  |
|                              hvm.gz                                    |
|                              pc.gz                                     |
|                              qsh.gz                                    |
|                              dossrv.gz                                 |
|                      l4/                                               |
|                              kickstart.gz                              |
|                              l4ka.gz                                   |
|                              sigma0.gz                                 |
|                                                                        |
|              src/     --- Source files                                 |
|                      Makefie  --- just "make clean; make"              |
|                      Maketool                                          |
|                      9/                                                |
|                              hvm/                                      |
|                              pc/                                       |
|                              port/                                     |
|                              ip/                                       |
|                              qsh/                                      |
|                                                                        |
|                      cmd/  --- Command source files                    |
|                              ext2fs/                                   |
|                              ....                                      |
|                              simple/                                   |
|                              test/                                     |
|                              usb/                                      |
|                                                                        |
|                      lib*/                                             |
*-------------------------------------------------------------------------
```

19.2　Make

(1) Makefile

Makefile の定義は、大きな抜けがあります (手が回らない)。特に ヘッダファイル (*.h) の更新に対しては無対処です。ヘッダファイルを修正した場合は、make clean してから make してください。

(2) システム全部の make

1. Chnge dir onto LP49-yymmdd/src/ directory

2. Clean up

3. Then make

   Many warnings...

4. 構築されたバイナリを、 ”LP49-yymmdd/rootfs/boot, bin/”ディレクトリにコピーする。

   LP49-yymmdd/rootfs/boot/* には、OS 本体が

   LP49-yymmdd/rootfs/bin/* にはコマンド類がコピーされる。

5. Linux の mkisofs コマンドを使って rootfs/ から CD ブートファイルイメージを作る

   LP49-yymmdd/mkcd を参照

```
[Linux]  mkisofs -R -b boot/grub/stage2_eltorito -no-emul-boot \
      -boot-load-size 4 -boot-info-table -o ../lp49-boot.cd  rootfs
```

19.3 ブート CD、ブートフロッピーのイメージファイル

lp49-boot.fd : 1.44 MB floppy image
lp49-boot-x2.fd : 2.88 MB floppy image
(1) 内容

```
lp49-boot.cd:
        l4/
                kickstart-0804.gz    --- L4 kernel startup
                l4ka-0804.gz         --- L4 micro kernel
                sigma0-0804.gz       --- L4 root pager
        boot/
                dossrv.gz    --- DOS file server
                hvm.gz       --- (not) Hypervisory module
                pc.gz        --- LP49 core process
                qsh.gz       --- LP49 quasi (debugging) shell
                grub/
                        menu.lst  --- Grub menu
                        stage1    --- Grub stage-1
                        stage2    --- Grub stage-2
        bin/
                cmd.zip    ---- ZIP file of LP49 commands.
                各コマンド
```

(2) When LP49 startup, boot file is mounted at the /t directory.
LP49[/]: ls /t

19.4 デバグ支援

(1) LP49/include/l4/_dbgpr.h ファイルは、デバッグ用機能を定義している。

```
extern int  l4printf_b(const char*, ...);
extern char l4printgetc();

#define  DBGPRN  if(_DBGFLG) l4printf_b
#define  DBGBRK  if(_DBGFLG) l4printgetc
#define  DD(x)   if(_DBGFLG) l4printf_b("%s: %x ￥n", __FUNCTION__, x);

#define  DPR     l4printf_r
#define  DPG     l4printf_g
#define  DPB     l4printf_b
```

(2) _dbgpr.h の使い方

```
 #define   _DBGFLG    1      //OR*  static int _DBGFLG = 1;
 #include  "_dbgpr.h"
_DBGFLG == 1  ==> Enable debug prints
_DBGFLG == 0  ==> Disable debug prints
```

(3) QSH の ”d” コマンド

ソースプログラムは、src/9/port/_dumptbl.c

19.5 実行時間計測

(check_clock() などの説明を付ける)

19.6 GCC について

GCC は version によって振る舞いが異なります。LP49 は FedoreCore Linux 搭載の GCC-3.2 と GCC-4.1 とでコンパイルをしている。GCC コンパイラによっては、make 時に__stack_chk_fail が未定義というエラーに陥るもしれない。GCC-4 からは、スタック上に確保した配列のオーバフローを実行時に検出するスタック保護の仕組みを取り入れたからである。このエラーが出た場合は、とりあえづ src/Maketool の CC の定義に–nostack-protector を追加してみて下さい。

```
CC = gcc --nostack-protector
```

ただし、このオプションは gcc-3.* では理解できずエラーになる。Makefile の中でコンパイラの version をチェックして、オプションの着脱をおこなうべきである。