Along the GNU/Hurd RPC way

A starting guide to contributing to the GNU Hurd

Samuel Thibault

2015 February 1st

It's all about freedom #0

“The freedom to run the program, for any purpose”

I.e.:

- Freedom from sysadmin!
  - WTH is fdisk/mke2fs/... hidden in /sbin?
  - I should be able to just work with my disk/network access
- Freedom to innovate
  - Experimental filesystem, personal work-flow, new kind of process combination,...

- Also provide freedom from misbehaving programs and drivers

It's all about freedom #0

```
From: xxx <xxx@yyy.fr>

Subject: Network expertise

Date: Thu, 31 Jan 2013 12:37:34 +0100

[…] Would it be possible to route to my VPN the
traffic of only one application?
```

Actually, also well-known classical issue of full-VPN: traffic of the VPN itself shouldn't go through the VPN!

And yet, here root capabilities!!

Spoiler: Yes, GNU/Hurd can already do it. Without even asking root.

It's all about freedom #0

Extensibility for the user

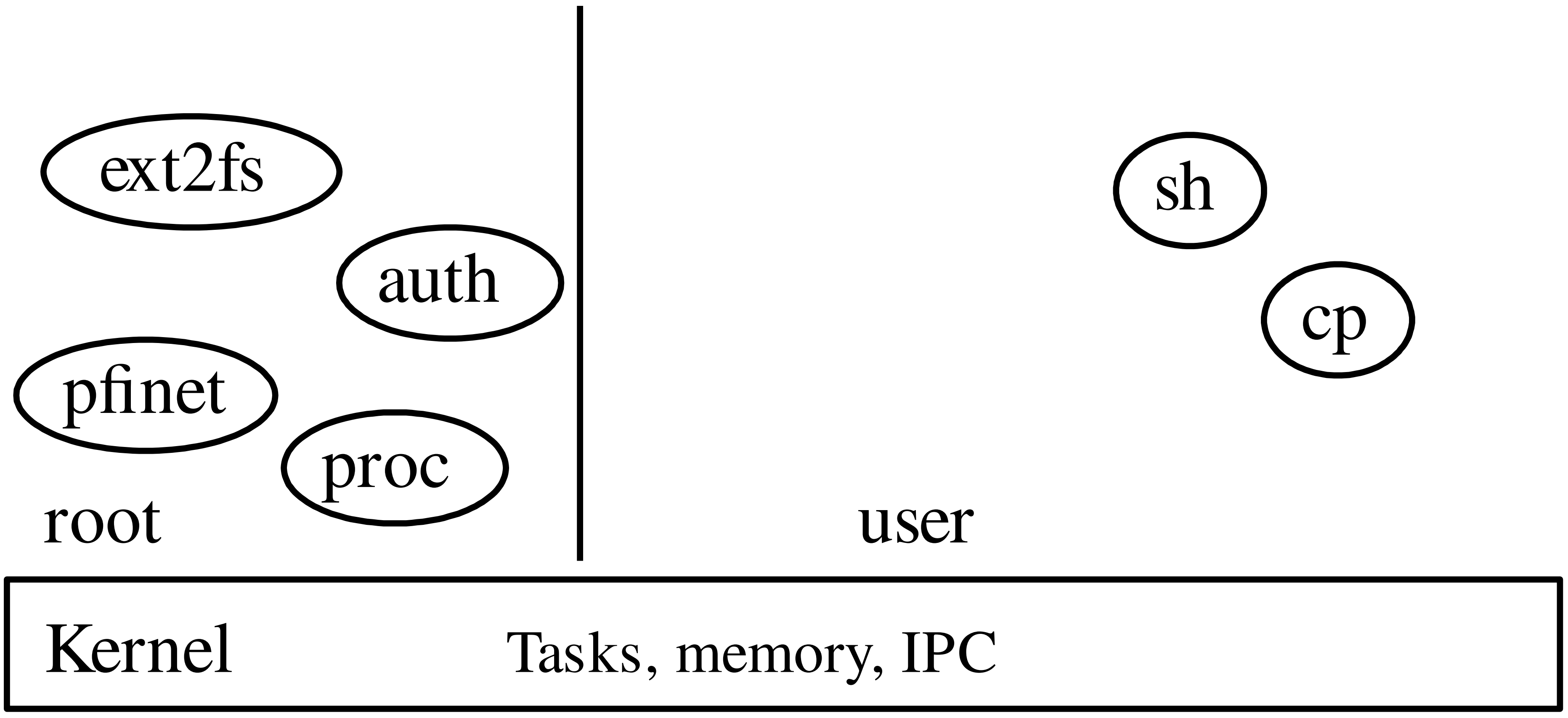
- Mount one's own files
  - Access archives content
  - Access remote files
  - Experiment with filesystems
- Access one's own network
  - Access remote networks / VPN
  - Access virtual machine network
- Redirect one's sound
  - Through network
  - Sound effects
  - Recording
- …
- and Flexible hardware support

Outline

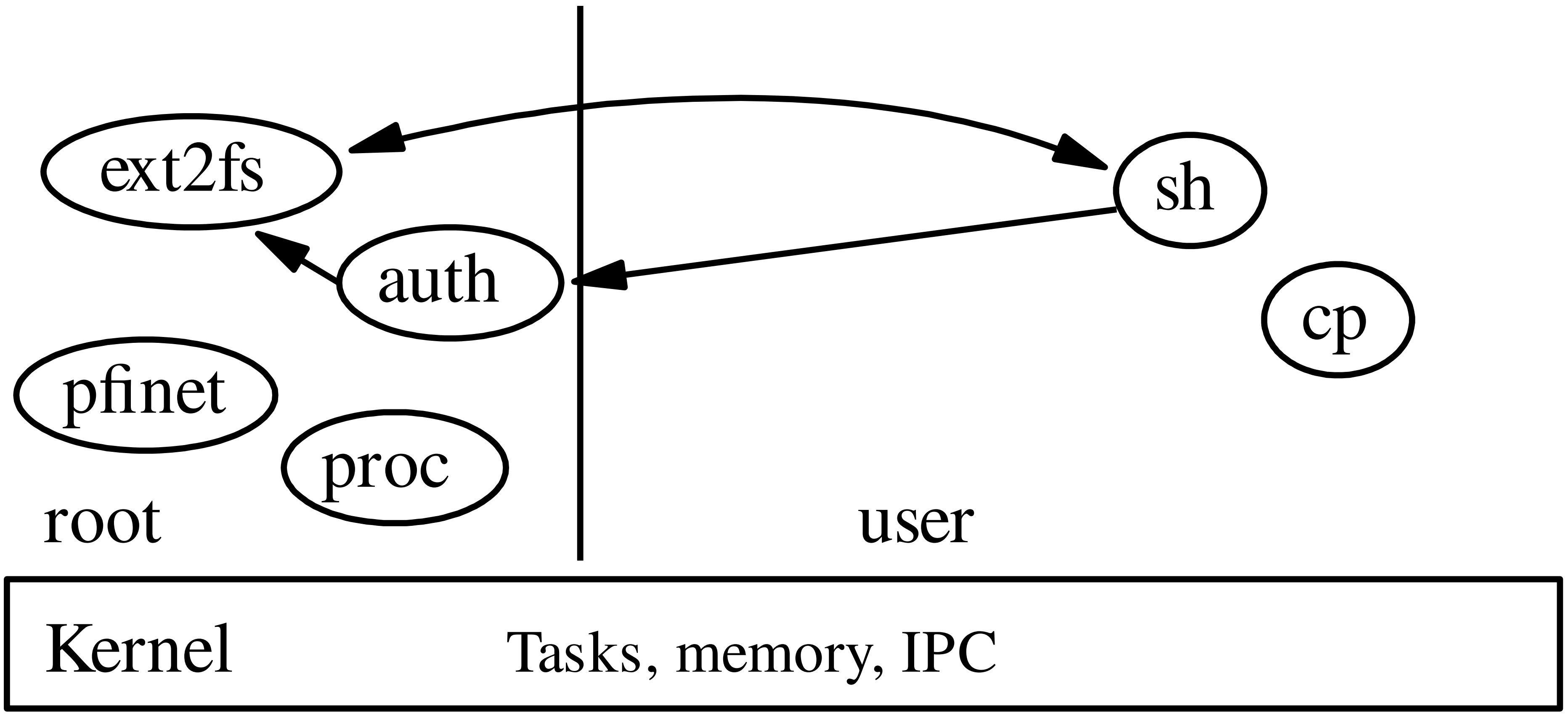
note: Start downloading glibc, hurd, gnumach source code now

- Hurd architecture Overview
- Flexibility, flexibility, flexibility!
- 3 Hurdish paths
  - ext2fs example
  - pflocal example
  - gnumach example
- Present & future

Micro-kernel layering



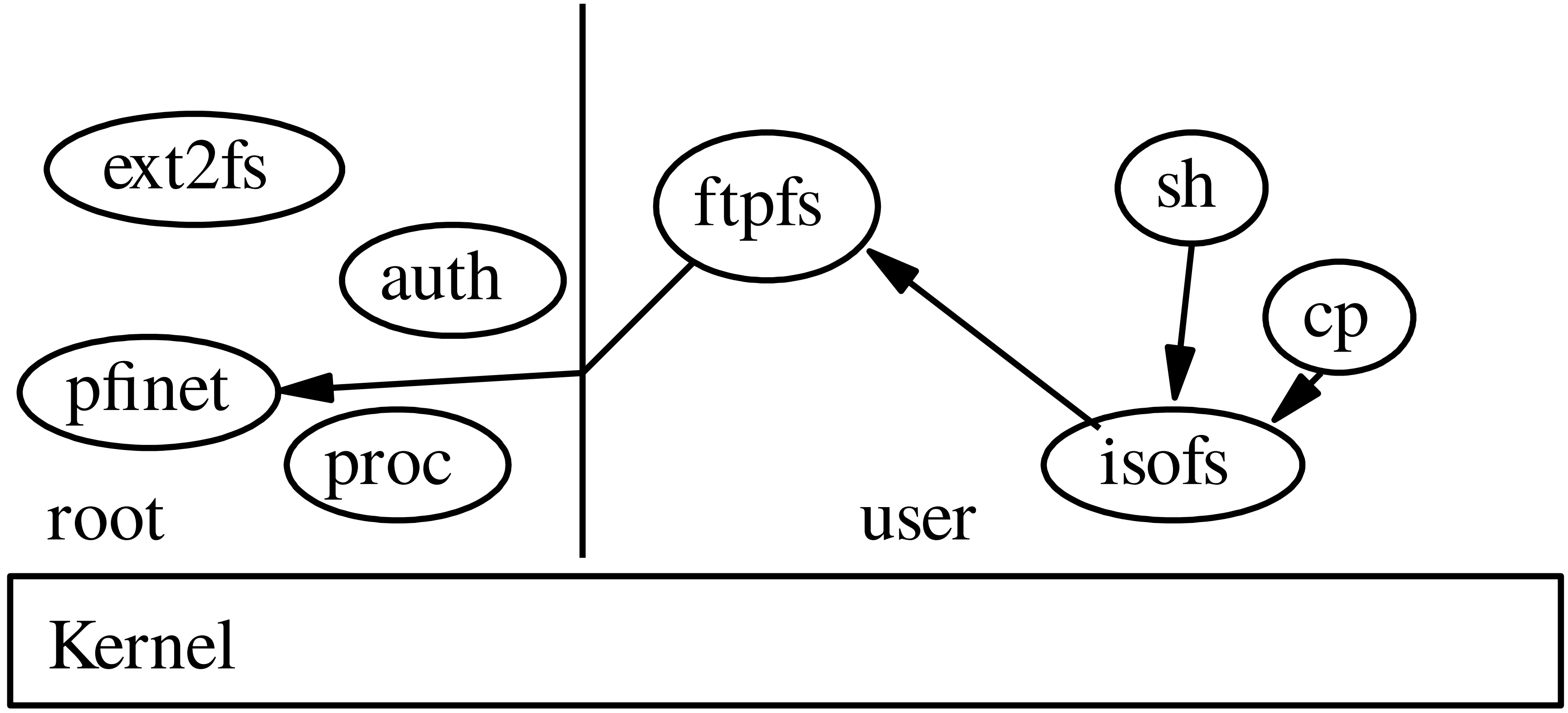
Micro-kernel layering



Micro-kernel layering

- Server crash? Not a problem
  - “Computer bought the farm” is just an error, not something-of-the-death
- Easier to debug/tune
  - Just run gdb, gprof, …
- Can dare crazy things
  - The Hurd console has dynamic font support
    - See chinese support in pseudo-graphical mode (actually pure VGA textmode!) of Debian installer.
- Kernel only handles Tasks, memory, IPC

Hurd possibilities



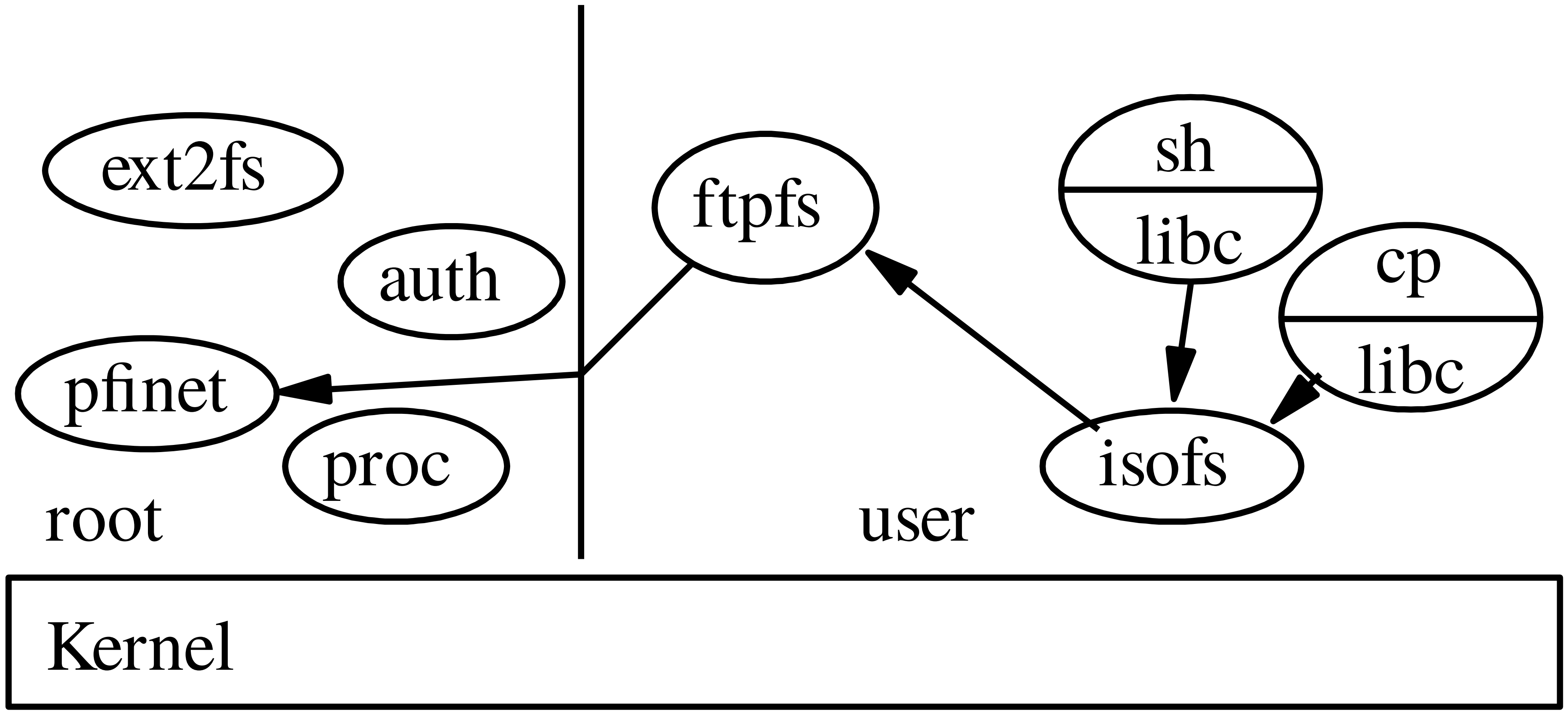
Hurd possibilities

```
€ settrans -c ~/ftp: /hurd/hostmux /hurd/ftpfs /
(just once for good)
€ settrans -a ~/mnt /hurd/iso9660fs
~/ftp://ftp.gnu.org/old-gnu/gnu-f2/hurd-F2-main.iso
€ ls ~/mnt
README-or-FAIL
…
```

- Only downloads what is needed.
- Can be permanently stored in ext2fs

```
€ settrans ~/.signature /hurd/run /usr/games/fortune
```

How does it work?



Rationale

- **Everything** is an (interposable) RPC
- Translators exposed in the FS

- The user gets to decide what/how to interpose
  - Without need for costly ptrace or fragile libc symbols interposition.
  - **Native** fakeroot/chroot
  - Fully virtualized and fine-grained interface
- Just need to use what's provided by the admin, e.g.
  - $HOME/
  - TCP/IP stack

and pile over it

Example: interpose TCP/IP stack

```
€ settrans -ca $HOME/servers/socket/2
   /hurd/pfinet -i $HOME/servers/tun0

€ openvpn … $HOME/servers/tun0 &

€ remap /servers/socket/2
        $HOME/servers/socket/2

€€€ wget www.gnu.org
```

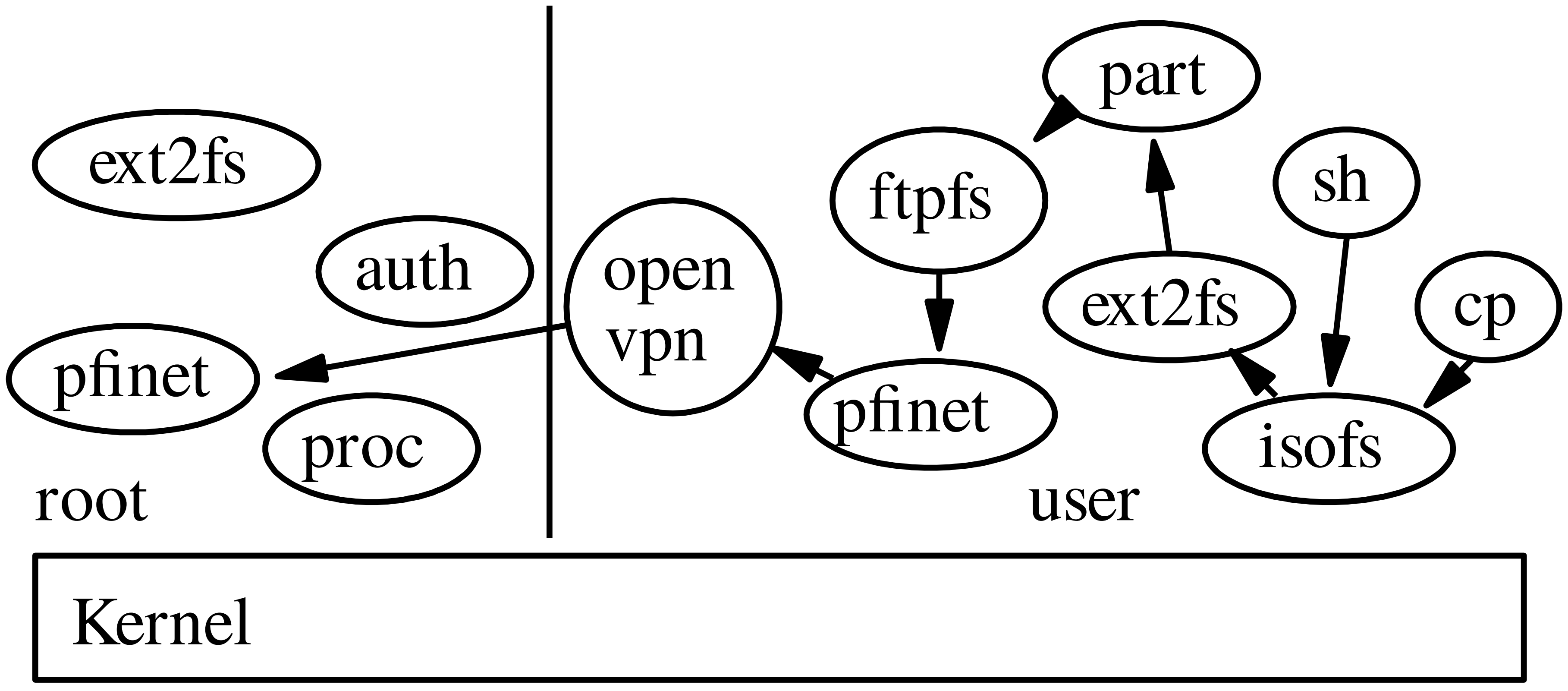
- My own translator
- Can plug my own VPN software
- Only wget accesses it (well, the shell too :))

But also

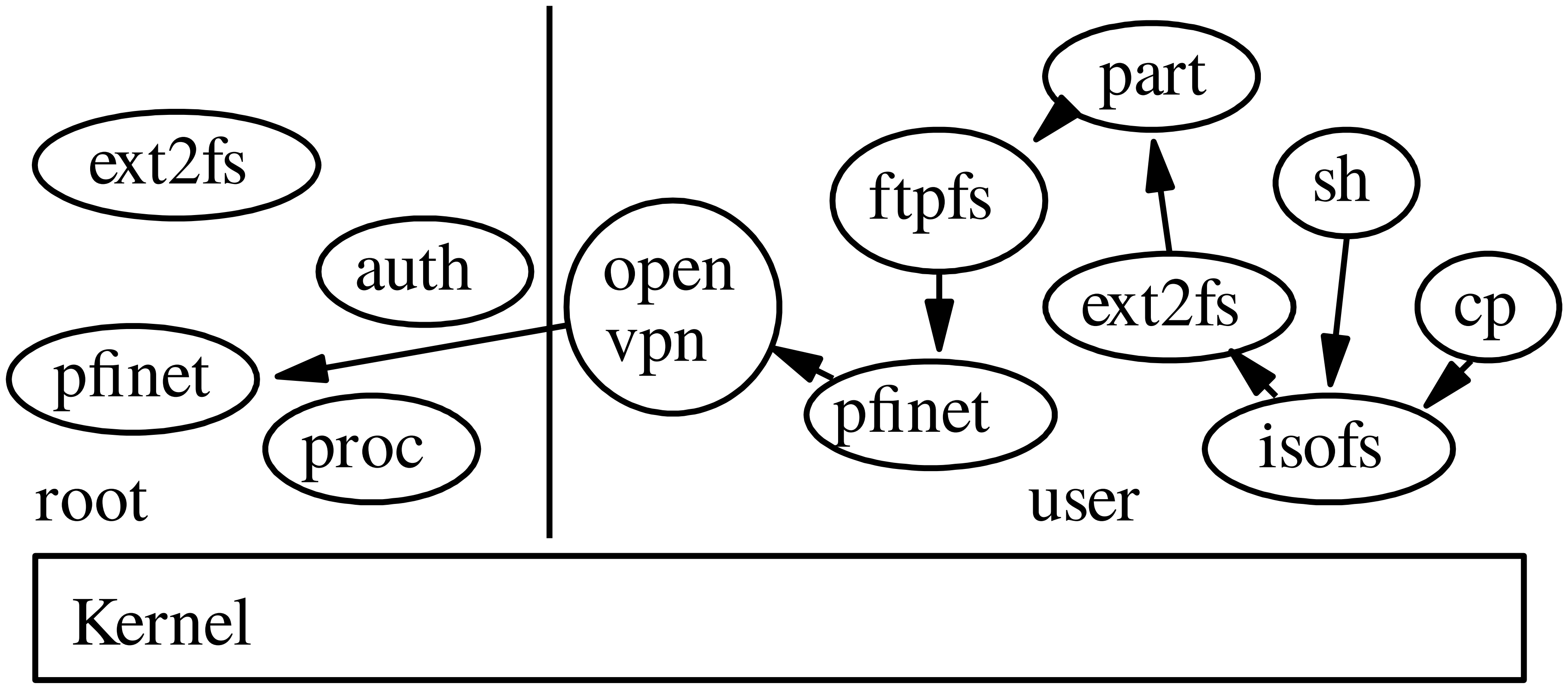
```
€ remap /bin/sh $HOME/bin/sh
€ remap /bin $HOME/unionbin
…
```

- Check out Stow/Nix/Guix!

Hurd possibilities (cont'ed)



Hurd possibilities (cont'ed)



i.e. ISO image inside a partitioned disk image on ftp over a VPN

Normal file path
ext2fs

ext2fs example

Bug report: “UTIME_NOW/OMIT are not defined” (for wine)

```
int fd;
struct timespec times[2];
fd = open (“foo.txt”, O_WRONLY);
times[0].tv_sec = time(NULL);
times[0].tv_nsec = 42424242;
times[1].tv_nsec = UTIME_OMIT;
futimens (fd, ts);
```

ext2fs example

RPC principle

- Open a connection
- Run RPCs over it
- Close the connection

Here,

- fd = open(“foo.txt”);
- futimens(fd);
- close(fd);

ext2fs example

Opening the connection

- foo.txt is a normal file

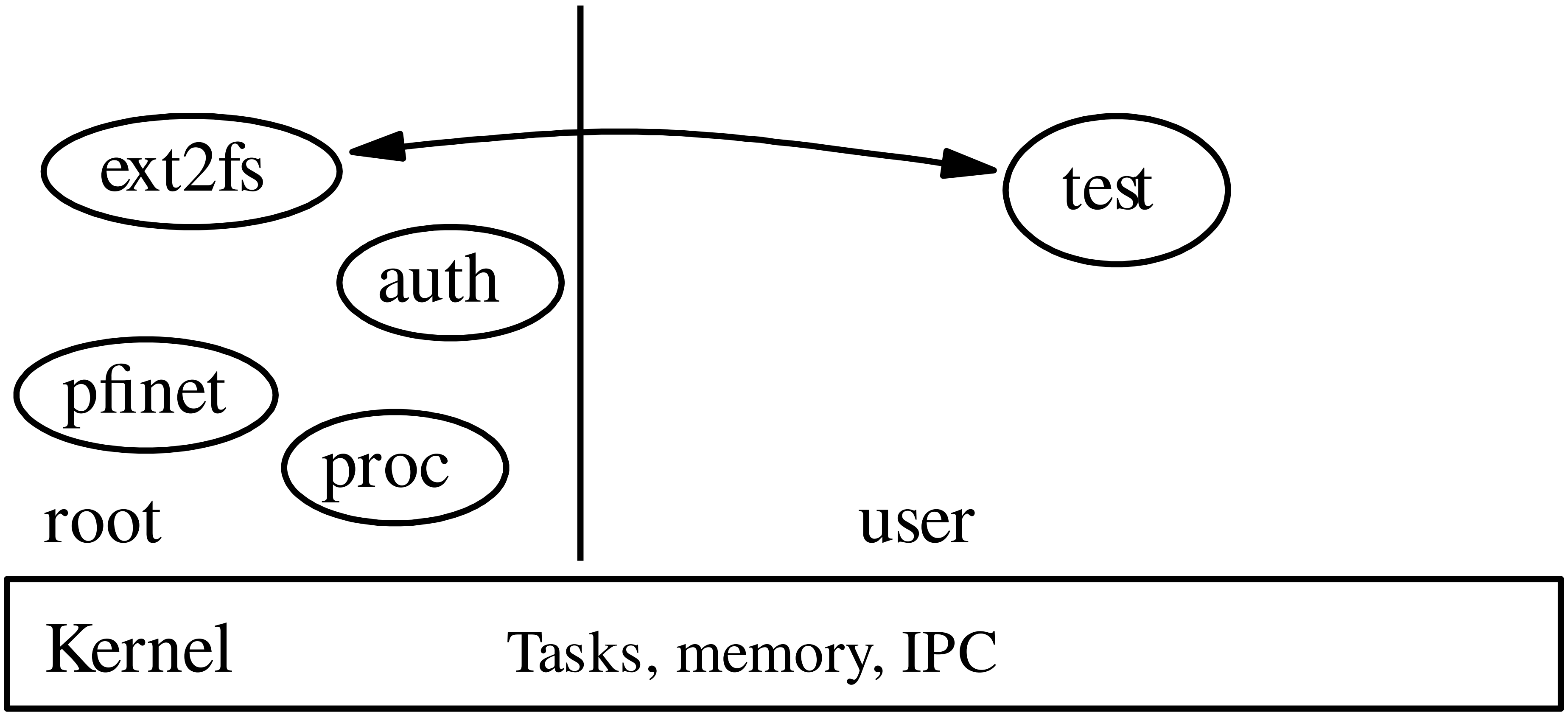
```
€ showtrans foo.txt
```

- foo.txt is in the current directory
- The current directory is served by ext2fs:

```
€ fsysopts .
ext2fs device:sd1
```

- So open(“foo.txt”) actually connects to ext2fs

ext2fs example



ext2fs example

RPC being run

- See futimens() source code in glibc

```
/usr/src/glibc€ find . -name futimens.c
./io/futimens.c
./sysdeps/mach/hurd/futimens.c ← that's it!
./sysdeps/unix/sysv/linux/futimens.c
```

- Basically just does

```
__file_utimes (port, atime, mtime);
```

- `port` is the low-level RPC port behind `fd`
- This is an RPC! Let's now look for the server side

ext2fs example

```
/usr/src/hurd€ rgrep file_utimes .
./hurd/fs.defs:routine file_utimes (
…
./libdiskfs/file-utimes.c:diskfs_S_file_utimes (struct protid...
./libtreefs/s-file.c:treefs_S_file_utimes (struct treefs_protid...
./libtrivfs/file-utimes.c:trivfs_S_file_utimes (struct trivfs_protid...
```

- but no ext2fs?!

```
/usr/src/hurd€ ldd /hurd/ext2fs
libdiskfs.so.0.3 => /lib/i386-gnu/libdiskfs.so.0.3 (0x01086000)
```

→ it's libdiskfs/file-utimes.c

ext2fs example

```
diskfs_S_file_utimes (... cred, ... atime, ... mtime) {
    …
    if (atime.microseconds == -1)
    …
    else {
        np->dn_stat.st_atim.tv_sec = atime.seconds;
        …
    }
    ...
```

So it's -1!

ext2fs example

Exercice for the audience

- Add #defines for UTIME_NOW and UTIME_OMIT to glibc/sysdeps/mach/hurd/bits/stat.h, see glibc/sysdeps/unix/sysv/linux/bits/stat.h for an example
- Add code to ./sysdeps/mach/hurd/futimens.c to handle the UTIME_NOW case.
- Add code to ./sysdeps/mach/hurd/futimens.c to put -1 in structure for the RPC in the UTIME_OMIT case.
- Test, enjoy, polish, submit!

socket path
pflocal

pflocal example

Bug report: “setsockopt(SO_SNDBUF) returns ENOPROTOOPT on PF_LOCAL sockets” (for globus-gram-job-manager)

```
int f;

int size = 1024;

f = socket (PF_LOCAL, SOCK_STREAM, 0);

if (setsockopt (f, SOL_SOCKET,
SO_SNDBUF, &size, sizeof (size)) < 0)

    perror ("setsockopt");
```

pflocal example

RPC principle

- Open a connection
- Run RPCs over it
- Close the connection

Here,

- fd = socket(PF_LOCAL);
- setsockopt(fd, SOL_SOCKET, SO_SNDBUF);
- close(fd);

pflocal example

What server will that be?

- ext2fs case was easy: file name, showtrans / fsysopts.
- socket case more involved
- socket()'s source code:

```
/usr/src/glibc€ find . -name socket.c
./socket/socket.c
./sysdeps/mach/hurd/socket.c
```

pflocal example

```
__socket (domain, type, protocol) {
    socket_t sock, server;
    server = _hurd_socket_server (domain, 0);
    __socket_create (server, type, protocol, &sock);
    return _hurd_intern_fd (sock, …);
}
```

socket_t is a typedef for the port type, so:

- get a port to a server
- RPC on it to get a port
- that will be the socket

pflocal example

```
hurd_socket_server (int domain, int dead) {
    … /* Code which basically does: */
    char name[sizeof (_SERVERS_SOCKET) + 100];
    sprintf (name, “%s/%d”, _SERVERS_SOCKET, domain);
    server = __file_name_lookup (name, 0, 0);
    return server;
}
```

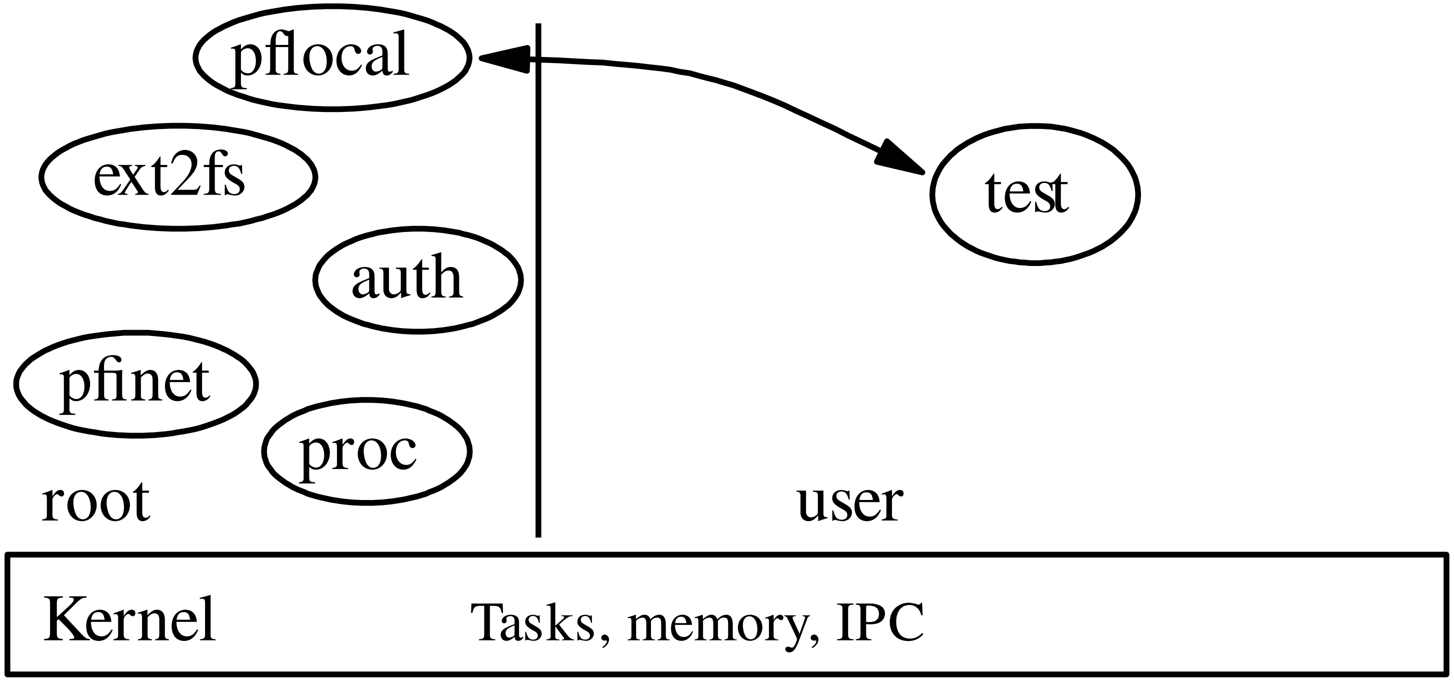
- _SERVERS_SOCKET is #defined to “/servers/socket”
- __file_name_lookup is what open() calls
- domain is PF_LOCAL, which is #defined to 1

→ it's merely opening /servers/socket/1

pflocal example

```
€ showtrans /servers/socket/1
/hurd/pflocal
```

→ it's translated by pflocal!



pflocal example

RPC being run

```
/usr/src/glibc€ find . -name setsockopt.c
./socket/setsockopt.c
./sysdeps/mach/hurd/setsockopt.c
```

- basically calls __socket_setopt on the port, i.e. an RPC

```
/usr/src/hurd€ grep -r socket_setopt .
./hurd/socket.defs:routine socket_setopt (
./pflocal/socket.c:S_socket_setopt (struct sock_user …
./pfinet/socket-ops.c:S_socket_setopt (struct sock_user …
```

pflocal example

```
S_socket_setopt (user, level, opt, value, value_len) {
    …
    switch (level)
        {
        default:
            ret = ENOPROTOOPT;
            break;
        }
    …
}
```

pflocal example

Exercice for the audience

- Add SOL_SOCKET and SO_SNDBUF cases in S_socket_setopt
- Notice that pflocal actually just uses libpipe for its buffering
- Find the “write_limit” buffer size in libpipes
- Implement there dynamically changing it
- Plug that into S_socket_set/getopt()
- Test, enjoy, polish, submit!

Memory management
gnumach

gnumach example

Bug report: “mlock() as non-root always returns EPERM” (for gnome-keyring)

```
    char s[128];
    if (mlock (&s, sizeof(s)) < 0)
        perror ("mlock");
```

```
/usr/src/glibc€ find . -name mlock.c
./misc/mlock.c
./sysdeps/mach/hurd/mlock.c
```

gnumach example

```
mlock (address, len) {
    mach_port_t hostpriv;
    __get_privileged_ports (&hostpriv, NULL);
    …
    __vm_wire (hostpriv, __mach_task_self(), page, len,
VM_PROT_READ);
    …
}
```

- __get_privileged_ports returns a port
- we make an RPC on it.

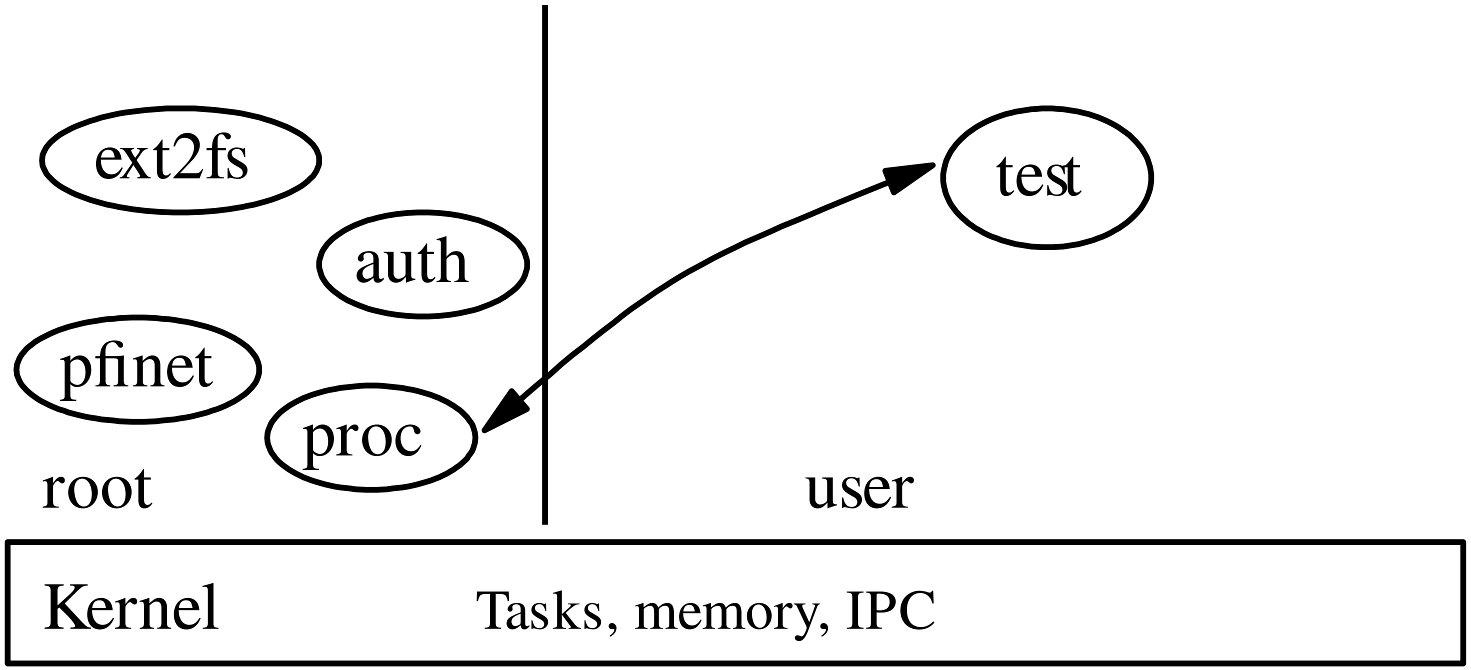
Hurd hackers know hostpriv is usually gnumach, but let's see how!

gnumach example

```
__get_privileged_ports (host_priv_ptr, device_master_ptr) {
    …
    __USEPORT (PROC, __proc_getprivports (port,
&_hurd_host_priv, &_hurd_device_master));
    …
    *host_priv_ptr = _hurd_host_priv;
    …
}
```

gnumach example

So we're actually first talking with the proc server



Because gnumach knows nothing about uids!

proc knows whether process is uid 0 and thus allowed to access the gnumach privileged port

gnumach example

```
/usr/src/hurd€ rgrep proc_getprivports
hurd/process.defs:routine proc_getprivports (
proc/host.c:S_proc_getprivports (struct proc *p,

S_proc_getprivports (p, hostpriv, devpriv) {
    if (! check_uid (p, 0))
        return EPERM;
    *hostpriv = _hurd_host_priv;
    *devpriv = _hurd_device_master;
    return 0;
}
```

gnumach example

proc

- started at system bootstrap
- passed the privileged port at that time
- checks uid

Why doing that way?

- Consider a sub-hurd
  - Control whether processes there can mlock()

gnumach example

Ok, so it's a gnumach RPC

```
/usr/src/gnumach€ rgrep vm_wire
./include/mach/mach_host.defs:routine  vm_wire(
./vm/vm_user.c:kern_return_t vm_wire(host, map, start, size, …

vm_wire (host, map, start, size, access) {
    if (host == HOST_NULL)
        return KERN_INVALID_HOST;
    …
    return vm_map_pageable_user (...);
}
```

gnumach example

So, what do we need to do?

- gnumach controls whether allowed or not.
- Don't want to let all processes mlock() a lot of memory
- `ulimit -l`, i.e. `setrlimit (RLIMIT_MEMLOCK)` controls how much is allowed for non-root, 64K by default
- Ideally, plug glibc's setrlimit() with gnumach
- As a first step, default to 64K

gnumach example

Exercise for the audience

- Add per-task vm_wire() counter to gnumach
- Allow tasks passing host == NULL to vm_wire() as much as 64K
- Patch setrlimit to advertise 64K as being fixed (for now).
- Test, enjoy, polish, submit!

State, news, future, etc.

Current State

Hardware support

- i686
- start of 64bit support
    - Kernel boots completely, now missing RPC 32/64bit translation
- DDE Linux 2.6.32 drivers layer for network boards
    - In userland netdde translator!
- IDE, Xorg, …
- AHCI driver for SATA (up to 2TiB disk support btw)
- Xen PV domU
    - Required GNU Mach changes only
- No USB, no sound yet

Current State

Software support

- Quite stable
  - Have not reinstalled boxes for years.
  - Debian buildds keep building packages, no hang after weeks!
- ~81% of Debian archive builds out of tree
  - XFCE, almost gnome, almost KDE
  - Firefox (aka iceweasel), gnumeric, …
- Standard *native* Debian Installer

Recent work

Special thanks to Justus Winter!!

- Init system decoupled
  - Allows to use standard Debian sysvinit scripts!
  - Using dmd for Guix & such
- Distributed mtab translator
- Various optimizations
  - Protected payloads
  - Lockless implementations
  - Paging management
  - Message dispatch
- Valgrind start-of-port

Releases

- Nice 0.401 release on April 2011.
- Arch Hurd LiveCD release on August 2011.
- Released Debian-unofficial wheezy/sid snapshot CDs on May 2013 \o/
- Hurd 0.5 released on 2013 Sept 27th \O/
  - Just in time for GNU's 30th birthday!
- Will soon release Debian-unofficial jessie/sid snapshot CDs

Future work

- Xen PVH support, X86_64 support
- Language bindings for translators (ADA?)
- Read-ahead
- {hdd,sound,usb}dde?
- GNU system: Guix/Hurd?
- Startup in scheme?
- Rump drivers?
- Your own pet project?

Thanks!

- http://hurd.gnu.org/
- http://www.debian.org/ports/hurd/
- http://people.debian.org/~mbanck/debian-hurd.pdf
- The increasing irrelevance of IPC performance for microkernel-based Operating Systems

  http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.37.9653&rep=rep1&type=pdf