not/configure

portability without pain

Alternative universes

What and why?

Plan 9 and **Inferno**

operating systems and supporting environments

distributed system

- collection of specialised services to build one system
- securely
- *portably*
- simply?

more than just a language

Plan 9

- a complete operating system
    - kernel and user processes
    - virtual memory
    - networking
    - graphics
    - applications (shell, editor, dev. tools, window system)
- distributed system
    - terminals, cpu server, file server

Inferno

Plan 9 ideas
Limbo (safe concurrent language, processes+channels)
Dis virtual (abstract) machine, with JIT
public-key authentication
native on ARM, PowerPC, x86
hosted on Linux, Windows, OS/X …

- looks like Inferno native OS to Inferno application
- looks like application to host OS
- includes /**net** interface
- cheaper than VMM

originally designed and used for embedded devices in distributed system

Plan 9 portability

mix heterogeneous hardware transparently

common file structures for distributed systems

all software is intended portable by design

- libraries
- compilers
- debuggers
- commands
- kernels

many architectures: x86, amd64, ARM, PowerPC, MIPS, SPARC, …

alt.universe

Design: name space, distribution, concurrency, heterogeneous

System calls: about 30

Libraries: libc, bio, thread, sec, auth, regexp, mach

Tools: C compiler, mk

Protocols: 9p, network independence, no sockets

Commands: rc, cpu, rio, mk, acme, sam, db, acid, bind, import, 8a, 8c, 8l, …

Unixy: cat, ed, ls, sed, sort, uniq. Utf8 throughout

Services & resources: file servers … no X11

System organisation

resources as ‘files’

computable name space

file service protocol (9P)

Distributed system implementation

serve tree using 9P on network file descriptor

import 9P (mount 9P connection in name space)

- network graphics? import /dev/draw
- network audio? import /dev/audio
- network gateway? import /net (or just /net/tcp)

cpu service (connect, export/import, bind)

either side, or both, can be file server

Concurrency

- designed for concurrency
  - symmetric multiprocessing
  - user-mode concurrency
- shared memory and channels

- most non-trivial file servers are concurrent programs
  - *exportfs, rio, acme, dns, cs, fossil, venti*

- window system (*rio*) and editor (*acme*) are concurrent programs and file servers

real-time support (EDF scheduler)

Plan 9 system interfaces

open, read/write, close
dup, pipe, fd2path, seek
create, remove, stat, wstat

bind, mount, unmount

rfork, wait, exits, exec
rendezvous, semacquire/semrelease
alarm/sleep, notify/noted
segbrk, segattach, segdetach, ...
errstr (*last error, as string*)

File servers: examples

- kernel services
  - /dev
    - mainly device drivers, union mounted
    - **data**, **ctl**
    - cons, consctl
    - audio, audioctl
    - eia0, eia0ctl
  - multiplexers are file trees
- user-mode services
- boring: dossrv, 9660srv, tarfs, …, ftpfs, nntpfs, paqfs

File servers: examples

- more interesting
    - dns
    - cs [recipes]
    - upas/fs
    - keyfs
    - factotum [fgui]
    - draw, rio
    - acme
    - plumber
    - fossil
    - iostats
- caches (9P ↔ 9P)
    - cfs (1:1)
    - fscfs (many:1)

Name spaces: *computable* name spaces

mount connection to file server on existing name

bind existing name over another existing name (alias)

unmount a connection or alias

union mounts

- /**bin** (search path)
- /**dev** (many devices)
- /**net** (many interfaces and protocols)

naming *conventions*

per-process granularity (no restrictions)

Networks

```
/net/
    arp
    cs
    dns
    ether0/
        addr
        clone
        ifstats
        stats
        0/
            ctl
            data
            ifstats
            stats
            type
        ...
    ether1/
        ...
```

```
/net/
    ipifc/
        clone
        stats
        0/
            ctl
            data
            err
            listen
            local
            remote
            snoop
            status
        ...
    iproute
```

```
/net/
    tcp/
        clone
        stats
        0/
            ctl
            data
            err
            listen
            local
            remote
            status
        1/
            ...
        ...
    ...
```

9P

file service protocol (RPC style, concurrent requests)

allows user-mode programs to create and serve trees of names

serve 9P on a file descriptor (eg, pipe, network)

mount file descriptor at existing directory in name space

operations below that directory become messages on the file descriptor

Support for portability

compilers

tools

conventions

simplicity

restraint

portability: how?

essentials of good programming practice

- abstraction and encapsulation
- simplicity and correctness

abstract away from details

- byte ordering not visible internally
- hardware instructions

increasing abstraction

- storage management
- concurrency

“porting” or “portability” is just a particular case

portability: how much?

easier the more you port

move a coherent environment

- commands
- libraries and interfaces

compilers, programming environment, native OS

- the impulse to original Unix ports & others

Plan 9

Inferno

Plan9ports

Example: Plan 9

mix heterogeneous hardware transparently

- common file structures for distributed systems

all software is intended portable by design

  - libraries
  - compilers
  - debuggers
  - commands
  - kernels

- many architectures
- cross-compile on any for all
  - cd /sys/src; objtype=power mk install

the outer limit

easier the more you port? do the lot:
- architecture independent *applications*
    - machine-independent object files
    - virtual machine (not *necessary*)
- cross platform O/S environment
    - emulated
    - *and* native

universal abstract interface for hardware and OS
- Inferno!
- Java? (no: it's an older, more primitive approach)

hurdles

lies, damned lies, and processor documentation

avoidable ones (at present)

- object and executable file details
- compiler suite details, reliability and stability

techniques

#include

~~#ifdef~~   ~~volatile~~   ~~*(unsigned long)p~~

text interfaces (eg, ctl files not ioctl); error strings; uid/gid; UTF8

explicit binary encoding/decoding, byte at a time

mk parts list

/env/cputype, /env/objtype

/bin is empty: **bind /$objtype/bin /bin; bind -b $home/bin/$objtype /bin**

/$objtype/lib /sys/include /$objtype/include

well-defined and invariant environment; setjmp/longjmp

cross-compilation is fundamental

include files

/sys/include: everything is portable

/$objtype/include: machine-specific

```
  72 amd64/include/u.h
  30 amd64/include/ureg.h
 102 total
```

include files

one per library, specified order (man page), defined contents

```
#include <u.h>
#include <libc.h>
#include <auth.h>
#include <authsrv.h>
#include <mp.h>
#include <libsec.h>
#include <String.h>
#include <thread.h>
#include <fcall.h>
#include <9p.h>
```

compiler suite

compiler (binary format, abstract assembly language)

loader (linker), produces executable

assembler (front end for loader)

no *cc* command! letter per arch: .6, **.8**, .q, .v, … → 6.out, 8.out, …

each component stored in per-target directory in */sys/src/cmd* (*qa*, *qc*, *ql*)

C compiler has target-independent library (in *cc*), loader in */sys/src/cmd/ld*

libraries: libc, libmach

supporting tools are portable (given *libmach*): acid, db

compilation

cross-compilation? 8c(/sys/src/cmd/qc), run qc → powerpc

cross-compile on any for all

    one source tree:

    cd /sys/src; objtype=power mk install

or

    mk installall   →   for(objtype in $CPUS) mk install

compiler construction

cross-platform debugging

mkfiles

```
</$objtype/mkfile
BIN=/$objtype/bin

TARG=rio
OFILES=\
	rio.$O\
	data.$O\
	fsys.$O\
	scrl.$O\
	time.$O\
	util.$O\
	wctl.$O\
	wind.$O\
	xfid.$O\

HFILES=dat.h\
	fns.h\

</sys/src/cmd/mkone
```

/amd64/mkfile

```
</sys/src/mkfile.proto

CC=6c
LD=6l
O=6
AS=6a
```

/sys/src/mkfile.proto

```
#
# common mkfile parameters shared by all architectures
#

OS=5678qv
CPUS=arm amd64 arm64 386 power mips
CFLAGS=-FTVw
LEX=lex
YACC=yacc
MK=/bin/mk

…
```

standard mkfiles

```
 112 sys/src/cmd/mkfile
  46 sys/src/cmd/mklib
  77 sys/src/cmd/mkmany
  60 sys/src/cmd/mkone
  43 sys/src/cmd/mksyslib
 338 total
```

configuration

specification and abstraction

make a decision (change with time)

mkfile is parametrised: </$objtype/mkfile

source code is not (as such), hence no #ifdef

examples: /sys/src/mkfile, /sys/src/mkone, mkmany, mklib, mksyslib

Inferno’s mkfiles

```
    mkhost-$HOST
    mkfile-$HOST-$TARGET
    mkone-$SHELLTYPE  # sh, rc, nt
```

mkfile examples

target class of system (eg, Inferno: Posix, Windows, Plan 9, other …)

named and labelled

$cputype vs $objtype

port compiler, kernel cd /sys/src/; objtype=… mk install # installall

rc shell

mk

cross-platform: access remote /proc

Data representation

byte ordering: spell it out

```
    uchar *p = …;
    s = (p[1]<<8)|p[0]; /* little endian 16-bit value */
    s = (p[0]<<8)|p[1]; /* big endian value */
```

avoid **short** or **long** for external data:

```
    struct {
        uchar op[2];
        uchar id[4];
        …
    };
```

9P Protocol

size[4] Tversion tag[2] msize[4] version[s]
size[4] Rversion tag[2] msize[4] version[s]

size[4] Tauth tag[2] afid[4] uname[s] aname[s]
size[4] Rauth tag[2] aqid[13]

size[4] Tflush tag[2] oldtag[2]
size[4] Rflush tag[2]

size[4] Tattach tag[2] fid[4] afid[4] uname[s] aname[s]
size[4] Rattach tag[2] qid[13]

size[4] Twalk tag[2] fid[4] newfid[4] nwname[2] nwname*wname[s]
size[4] Rwalk tag[2] nwqid[2] nwqid*wqid[13]

size[4] Topen tag[2] fid[4] mode[1]
size[4] Ropen tag[2] qid[13] iounit[4]

size[4] Tcreate tag[2] fid[4] name[s] perm[4] mode[1]
size[4] Rcreate tag[2] qid[13] iounit[4]

size[4] Tread tag[2] fid[4] offset[8] count[4]
size[4] Rread tag[2] count[4] data[count]

size[4] Twrite tag[2] fid[4] offset[8] count[4] data[count]
size[4] Rwrite tag[2] count[4]

size[4] Tclunk tag[2] fid[4]
size[4] Rclunk tag[2]

size[4] Tremove tag[2] fid[4]
size[4] Rremove tag[2]

size[4] Tstat tag[2] fid[4]
size[4] Rstat tag[2] stat[n]

size[4] Twstat tag[2] fid[4] stat[n]
size[4] Rwstat tag[2]

size[4] Rerror tag[2] ename[s]

Support for portability

compilers

tools

conventions

simplicity

restraint

Native kernels

no need to rebuild the hardware in software

map the software requirements (interfaces) *into* the hardware

- the mapping need not be surjective!
- don't make hardware implementation visible needlessly

abstraction to hide details (eg, MMU implementation)

Network name resolution: domains

/net/dns

write name to be translated

read sequence of possible translations, one per line

```
> www.google.com

www.l.google.com ip     173.194.66.104
www.l.google.com ip     173.194.66.106
www.l.google.com ip     173.194.66.147
www.l.google.com ip     173.194.66.103
www.l.google.com ip     173.194.66.99
www.l.google.com ip     173.194.66.105

> google.com soa
google.com soa   ns1.google.com dns-admin.google.com 2012042000 7200 1800 1209600 300
```

Network name resolution: symbolic names

/net/cs

translates names for variety of networks and protocols

write network name to be translated

[*net* !] *netaddr* [! *svcname*]

read set of *recipes*, one per line

```
> net!dispensa!9fs

/net/il/clone 144.32.112.69!17008
/net/tcp/clone 144.32.112.69!564
```

network independent

telnet *net***!**host!svc   [text, /net/cs]

add pk network /net/pk/…

no change to source or executable