98-023A : Concurrent and Distributed Programming w/ Inferno and Limbo

Phillip Stanley-Marbell
pstanley@ece.cmu.edu

Lecture Outline

- Native Kernel Initialization

No Class Next Week

- Week 1: Introduction to Inferno
- Week 2: Overview of the Limbo programming language
- Week 3: Types in Limbo
- Week 4: Inferno Kernel Overview
- Week 5: Inferno Kernel Device Drivers
- Week 6: NO CLASS
- Week 7: C applications as resource servers: Built-in modules and device drivers

Spring Break

- Week 8: Case study I — building a distributed multi-processor simulator
- Week 9: Platform independent Interfaces: Limbo GUIs; Project Update
- Week 10: Programing with threads, CSP
- Week 11: Debugging concurrent programs; Promela and SPIN
- Week 12: Factotum, Secstore and Inferno's security architecture
- Week 13: Case study II — Edisong, a distributed audio synthesis and sequencing engine

Kernel Init and Startup : `l.s`

- First entry point is `/os/`*systemarch*`/l.s`
- l.$O must be the first item in OBJ list in mkfile
- l.s sets up some machine state, e.g. ensure CPU is in supervisor mode
- l.s calls kernel C startup code, main() in main.c
- Several more details in the case of x86 (e.g., the Plan 9 boot loader 9load sets up the MMU, real/vs protected mode dance, etc.)

Kernel Init and Startup : `main.c`

- Initial cleanup
  - e.g., Zero out uninitialized memory segment
  - Setup cache configuration
- `machinit()`
- `archreset()`
- `confinit()`
- `links()`
- `xinit()`
- `poolinit()`, `poolsizeinit()`
- `trapinit()`, `clockinit()`
- `procinit()`
- `chandevreset()`
- `userinit()`
- `schedinit()` (Initialization ends here: `schedinit()` never returns)

machinit()

- Clears the Mach structure (remember, last lecture ?)
  - Mach is defined in /os/*archname*/dat.h

```
struct Mach
{
      ulong    ticks;

      /* of the clock since boot time */
      Proc     *proc;

      /* current process on this processor */
      Label     sched;

      /* scheduler wakeup */
      Lock      alarmlock;

      /* access to alarm list */
      void      *alarm;

      /* alarms bound to this clock */
      int       machno;
      int       nrdy;
      int       stack[1];
};
```

`archreset()`

- In `/os/`*`archname`*`/arch`*`XYZ`*`.c`
- System architecture specific initialization
- Might not have to do anything
    - E.g., in the `ks32` port
- `/os/`*`archname`*`/arch`*`XYZ`*`.c` also contains code for other board/architecture specific operations

confinit()

- Does any architecture specific initialization
  - Calls archconfinit() from /os/*archname*/arch*XYZ*.c
- Sets up the Conf *conf structure
  - Conf structure is defined in /os/*archname*/dat.h
  - npage, base0, base1 setup by xinit()

```
struct Conf
{
        ulong     nmach; /* processors */
        ulong     nproc; /* processes */

        /* total physical pages of memory */
        ulong     npage0;
        ulong     npage1;

       /* highest physical address + 1 */
        ulong     topofmem;

        ulong     npage;
        ulong     base0;   /* base of bank 0 */
        ulong     base1;   /* base of bank 1 */

       /* max interrupt time allocation in bytes */
        ulong     ialloc;

        ulong     flashbase;
        ulong     cpuspeed;
        ulong     pagetable;
        int       useminicache;
        int      cansetbacklight;
        int      cansetcontrast;
        int      remaplo;
        int      textwrite;
};
```

`links()`

- This is defined in the C source generated by mkdevc, upon parsing the kernel config file
  - For all the entries in the links section, *`entryname`*`link()` is called
  - For example, for the following link section in a kernel config file:

```
link
    ether2114x     pci
    ps2mouse
    ethermedium
```

- The following code is generated (in *confname*`.c`) during the mk
- `void links(void){`
  - `ether2114xlink();`
  - `ps2mouselink();`
  - `ethermediumlink();`
- `}`

`xinit()`

- In `/os/port/xalloc.c`
- Sets up the `base` and `npage` variables in the `Conf` structure, i.e., sets up knowledge of memory
- Low-level memory allocation routines
  - `xalloxz()`
  - `xalloc()`
  - `xfree()`

`poolinit()`, `poolsizeinit()`

- In `/os/port/alloc.c`
- Memory in Inferno is managed as a set of fixed size "pools"
  - `main`
  - `heap`
  - `image`
  - E.g., memory for on-screen images is allocated from the `image` pool
- `poolsizeinit()` is in `main.c`
- Uses low-level memory allocation routines previously mentioned, from `/os/port/xalloc.`

`trapinit()`

- In `/os/`*`archname`*`/trap.c`
- Sets up exception stacks
- Installs interrupt handlers

`clockinit()`

- In `/os/`*`archname`*`/clock.c`
- Various routines for managing hardware timer
  - Enable timer
  - Disable timer
  - Get number of clock ticks since CPU initialized

procinit()

- In `/os/port/proc.c`
- Allocates memory for process list
  - `nproc` variable in the `Conf` structure

chandevreset()

- In `/os/port/chan.c`
- Calls the `dev`*XYZ*`reset()` routines of all device drivers
- Recall, `devtab[]` array in *`archname`*`.c`, generated during kernel compile by `mkdevlist`
  - `devtab[]` contains pointers to a `Dev` structure for each device driver
  - Recall that `Dev` structure for each device driver contains pointers to functions for initialization, and for handling local procedure call versions to Styx protocol

`userinit()`

- In `main.c`
- Creates the first system process, `init0()`, running as the user "`eve`"
- Marks this process as ready/runnable
- `init0()` calls `chandevinit()`
- `init0()` makes the Dis VM run the compiled Limbo program `osinit.dis`

`chandevinit()`

- In `/os/port/chan.c`
- Calls the `dev`*`XYZ`*`init()` routines of all device drivers (recall, we previously called their `reset()`s)
- Recall, `devtab[]` array in *`archname`*`.c`, generated during kernel compile by `mkdevlist`
  - `devtab[]` contains pointers to a `Dev` structure for each device driver

`schedinit()`

- In `/os/port/proc.c`
- This is the entry point for the scheduler
- `schedinit()` calls `sched()`
- Henceforth, processes run as scheduled by kernels scheduler (obviously)

Next

- C applications as Inferno resource servers : Built-in modules and device drivers
- No class next week (Feb 16, Feb 18)
- Homework 2 not due until Feb 23

Fin.