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

Phillip Stanley-Marbell
pstanley@ece.cmu.edu

Lecture Outline

- Limbo data types
- We'll spend the whole lecture solving all the exercises from chapter 2 and 3 of IPWL book

Course Outline : Syllabus

- 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

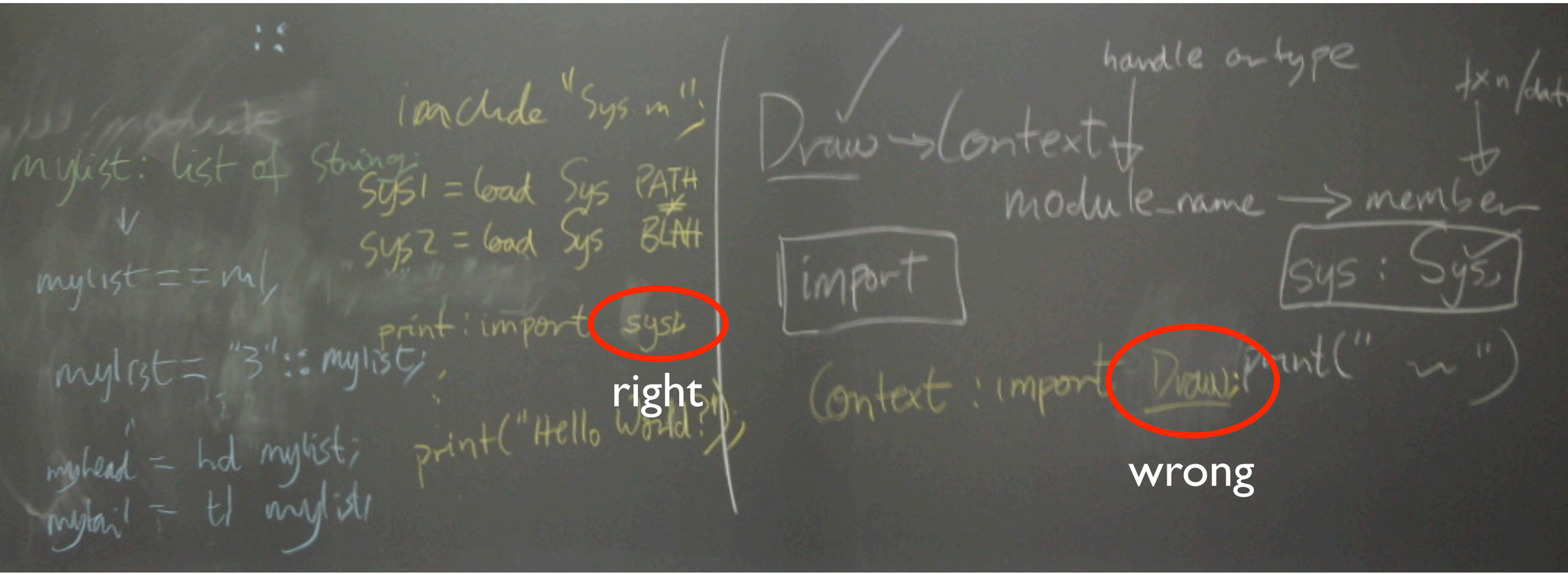
Inferno's VM: Dis

- Applications compiled for execution on the Dis VM
- Dis has a memory-to-memory architecture, optimized for on-the-fly compilation (contrast to the Java Virtual Machine's stack architecture)
- Many Dis VM opcodes map directly to Limbo language constructs, but can support other languages

Problem Solving/Demo:

Review — IPWL Chapter

2 Problems

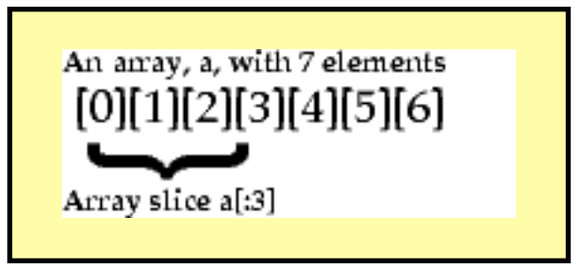

include "Sys.m";
mylist: list of string;
Sys1 = load Sys PATH
Sys2 = load Sys BLAH
mylist == nil
print: import sysL
mylist = "3" :: mylist;
myhead = hd mylist;
mytail = tl mylist
print("Hello World?")
right
Draw->Context
handle on type
module_name -> member
txn/out
import
sys : Sys;
Context : import Draw;
print("
wrong

Language Data Types

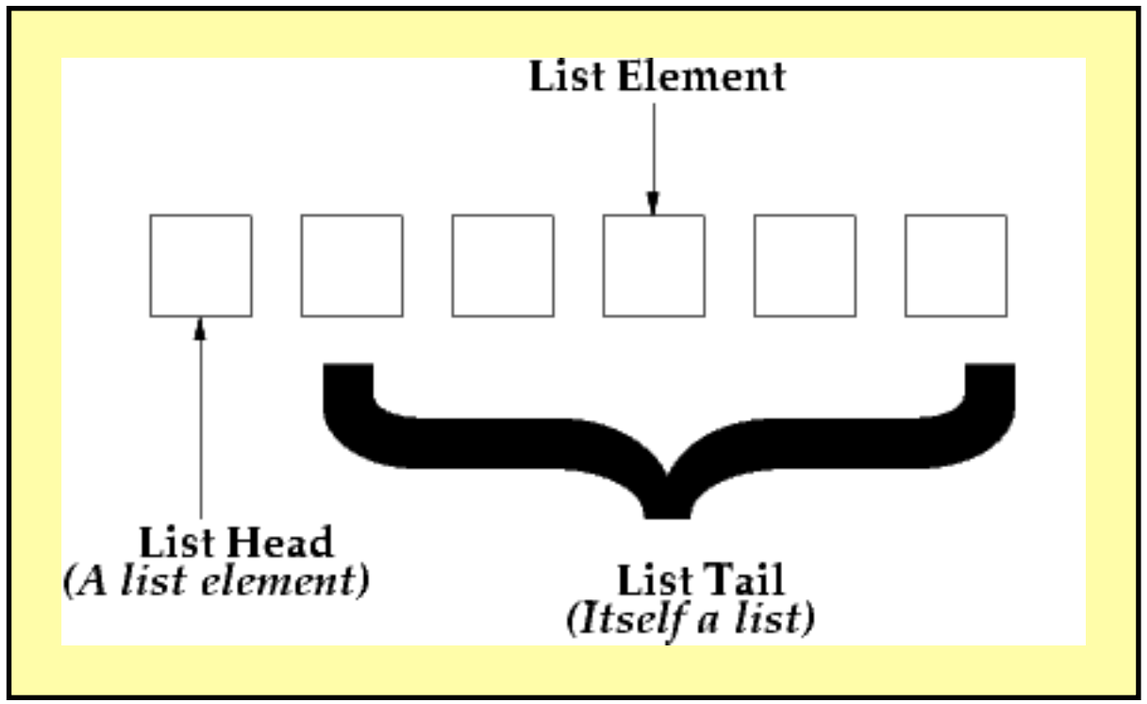
- Basic types
  - `int` — 32-bit, signed 2's complement notation
  - `big` — 64-bit, signed
  - `byte` — 8-bit, unsigned
  - `real` — 64-bit IEEE 754 long float
  - `string` — Sequence of 16-bit Unicode characters

- Structured Types
  - `array` — Array of basic or structured types
  - `adt`, `ref adt` — Grouping of data and functions
  - `list` — List of basic or structured data types, list of list, etc.
  - `chan`
  - Tuples

Arrays



Lists


List Element
List Head
(A list element)
List Tail
(Itself a list)

Problem Solving/Demo:
IPWL Chapter 3
Problems

Channels

- Channels are communication paths between threads
- Declared as `chan of` <any data type>
  - `mychan : chan of int;`
  - `somechan : chan of (int, string, chan of MyAdt);`

- Synchronous (blocking/rendezvous) communication between threads
- Channel operations
  - Send : `mychan <-= 5;`
  - Receive : `myadt = <- somechan;`
  - Alternate (monitor multiple channels for the capability to send or receive)

Example *(what does it do ?)*

```
implement X;
...
init(nil : ref Draw->Context, nil : list of string)
{
        sys = load Sys Sys->PATH;

        i := 2;
        sourcechan := chan of int;
        spawn sieve(i, sourcechan);
        while () sourcechan <-= i++;
}

sieve(ourprime : int, inchan : chan of int)
{
        n : int;
        sys->print("%d ", ourprime);
        newchan := chan of int;

        while (!((n = <-inchan) % ourprime)) ;

        spawn sieve(n, newchan);
        while ()
        {
                if ((n = <-inchan) % ourprime)
                {
                        newchan <-= n;
                }
        }
}
```


O/P
2
3

Next week

- The Dis VM and module binary format
- Limbo data types and the Dis VM

Fin.