Phony Programming
(Series 60 Symbian Phones)

Larry Rudolph
MIT 6.883 & SMA 5508
March 6, 2006

What's so special about phones?

- Ignorant Operator
  - really ignorant operator
- Scarce Resources
  - Power -- slow processor, small memory, small disk
  - Size -- Screen and keyboard area
  - Heat -- To keep this low, must constrain above
  - Price -- Different than PC, since each feature costs the same, due to volumes
  - Reboot -- these are rare events
  - Brand loyalty -- hardly any, mistakes are costly
- IT'S NEW AND EXCITING!

What's so special about phony programming?

- Inherently very little --- Practically very much
- History
  - originally: phones were h/w appliances
  - past: s/w - h/w co-design reduced costs
  - near past: new features all s/w
  - near future: 3rd party software
- In computer world, two approaches
  - Intel: H/W, Microsoft: O/S, separate h/w from s/w
  - Apple: H/W & S/W, easier to integrate, high costs $
- Companies have no history of being open !!

Massachusetts Institute of Technology

Current Choices

- Most phones have two ASICs
  - Modem/Telephone service
  - Embedded processor and extra features
- Palm-Based: Slow to integrate
- Pocket-PC: Appears to be aggressive
  - Can they maintain two OS's (Windows & WinCE)?
- Linux: (actually, Linux+Java)
  - Still hidden from user Motorola phones
  - iPaq 6315 has GSM/SIM
- Symbian: Mostly on Nokia and Sony/Ericsson
- IDC Predicts: Pa:10%, Po:27%, Li:4%, Sy:53%

Palm Phones

Pocket PC Phones

Linux Based Phones

Symbian Phones

Symbian Epoc OS

- Originally developed for the Psion handheld computer
  - competition with Palm
  - single user, small memory, instant-on, no network
- EPOC operating system
- Symbian independent company
  - partly owned by Nokia, Sony (no one controls them)
  - EPOC and Symbian names became intermixed
- Nearly all documentation and tools are for commercial developers
  - high start-up cost

CSAIL

Larry’s Deja Vu

- Despite the fact that
  - Programmable Mobile Phones are new artifacts
  - the OS and programming repeat mistakes
- Designs that make sense for disconnected, single use devices, remain as devices become connected and multi-use and multi-tasked.
  - Inertia is a powerful force

Massachusetts Institute of Technology

Programming Languages Series 60 Phones

- C++ for Symbian
  - access to all the phones functions
  - not so easy
- Java
  - highly sandboxed.
  - no access to file system, phone, and more
  - not the choice language for virus writers
- Python
  - nearly ready for public release
  - will have interface to all Symbian API's

Symbian OS Basics

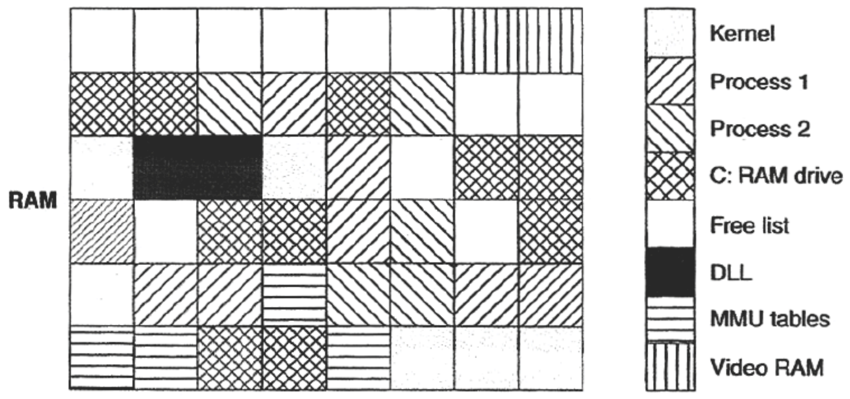
- Kernel: Protected Mode; Controls H/W
- Server: Manages one or more resources
  - no UI (user interface)
  - yes API (application program interface)
  - may be device driver or kernel service
- Application: A program with a UI
  - Each application is a process; own virtual address
  - If interacts with server, can be called a client
- Engine: part of app. manipulates its data not UI

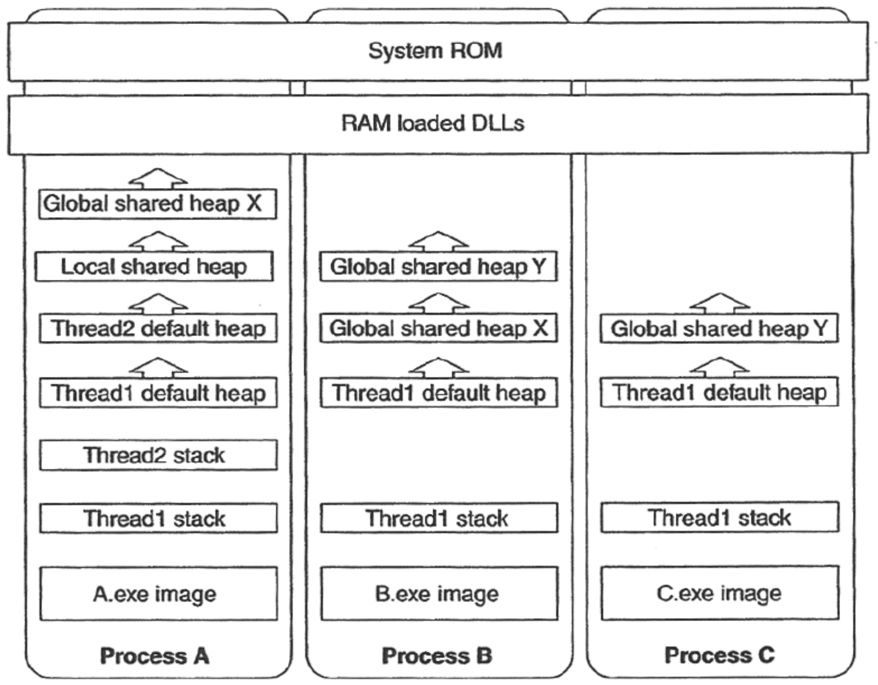
Processes & Threads

- Three boundaries
  - DLL or module: cheap to cross
  - Privilege: medium to cross
  - Process: expensive to cross
- Processes, Threads, Context Switches
  - Process: has its own address space (256 MB)
  - Thread: Unit of execution within a process
    - Preemptively scheduled; 2MB nonshared ==> 128 thds/pro
- Executables
  - exe: single entry point
  - dynamic link library (DLL): multiple entries
    - Shared (OS) vs Polymorphic (app)

Massachusetts Institute of Technology

RAM Memory Parts





Memory

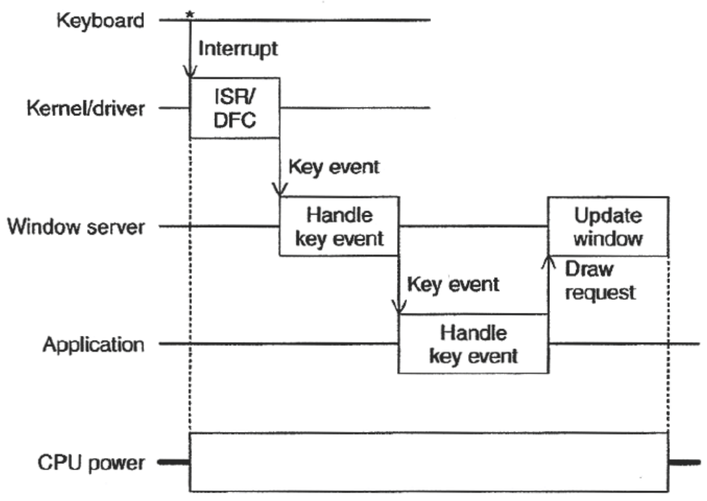
- RAM: partitioned into 4k pages
  - kernel, process, DLL, MMU tables, Video, C:
  - Thread memory: Shared:Heap, Nonshared:Stack
- DLL -- no writable static data (yes for exe's)
  - requires multiple copies of instantiated dll
- Files (does this remind you of something)
  - C: RAM -- r/w file system. Zero'd on cold boot
    - restored from ROM on cold boot
  - Z: ROM -- can be reflashed (not easy)
  - D: Memory Card
    - 512 byte blocks written atomically; VFAT format

Event Handling

- Efficient handling major OS design
    - native OS server is single event-handling thread
- Symbian organized as Event-driven
- Active Objects: non-preemptive event handling
    - and client-server structure

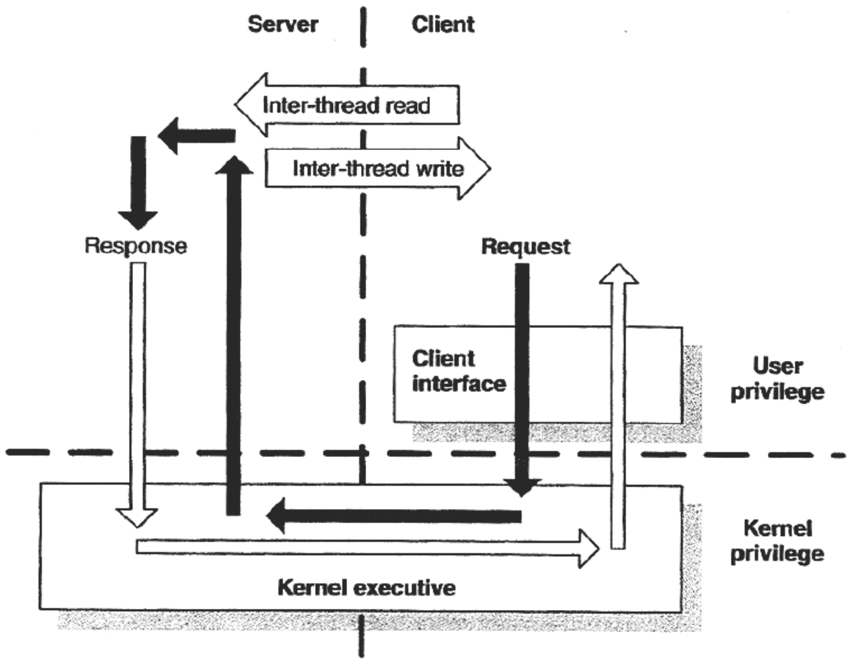
Massachusetts Institute of Technology

Response to key-press



Active Objects

- Each active object has virtual member function called **`RunL()`**
  - gets called when event happens
  - events have associated priority
- Active Objects execute **non-preemptively**
  - RunL() should execute for short time
  - no need for mutex, semaphores, critical sections
  - fewer context-switches
- Compute-intensive threads:
  - Simulate using pseudoevents
  - split task into pieces, generate low-prio event



Java Programming

- Lots of examples at www.forum.nokia.com
  - great for games and network connectivity
- Compile on server
- Install on phone via:
  - bluetooth connection
  - mms message (email)
  - upload from web server
- On phone, java applet must be opened to install before being run.

CSAIL

C++ Programming

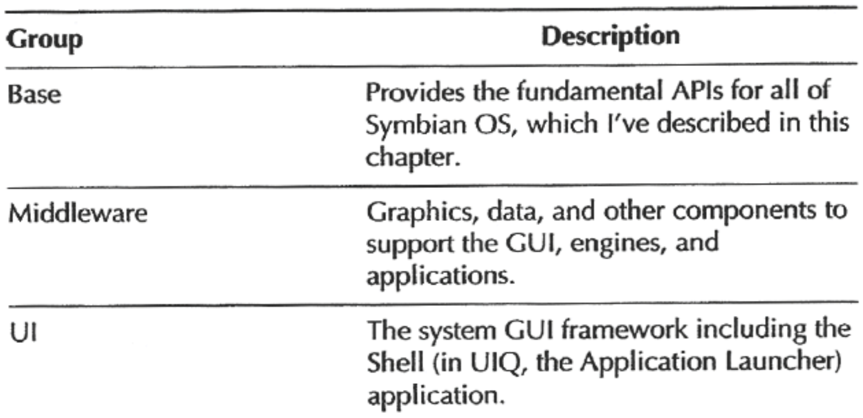
- The real stuff but documentation is difficult
- Everything depends on the SDK
  - Software Development Kit runs under Windows
  - The processor on phone is ARM (same as iPaq)
  - Must do cross-compilation
    - compile with different libraries and assembly instructions
- There is an “Emulator” -- but it is really a simulator
  - e.g. it executes x86 code not arm code
  - so must compile either for ARM or x86
  - emulator is needed for debugging

Massachusetts Institute of Technology

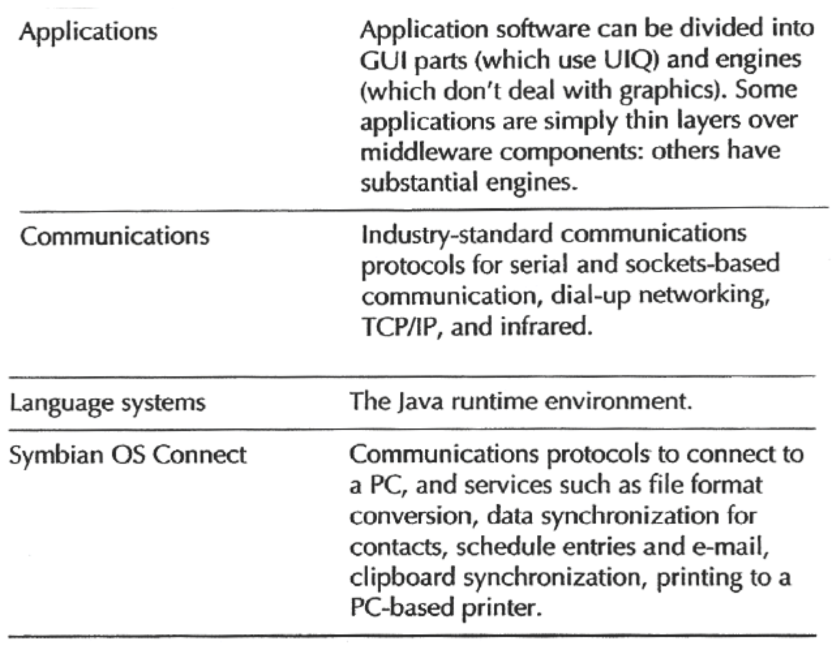
C++ Programming

- Tutorials: C++ for Java programmers
  - **http://www.cs.brown.edu/courses/cs123/javatoc.shtml**
  - There are many others on-line
  - Easier to go from java to c++ (java has less weirdness)

Symbian Layers



Group	Description
Base	Provides the fundamental APIs for all of Symbian OS, which I've described in this chapter.
Middleware	Graphics, data, and other components to support the GUI, engines, and applications.
UI	The system GUI framework including the Shell (in UIQ, the Application Launcher) application.



Applications	Application software can be divided into GUI parts (which use UIQ) and engines (which don't deal with graphics). Some applications are simply thin layers over middleware components: others have substantial engines.
Communications	Industry-standard communications protocols for serial and sockets-based communication, dial-up networking, TCP/IP, and infrared.
Language systems	The Java runtime environment.
Symbian OS Connect	Communications protocols to connect to a PC, and services such as file format conversion, data synchronization for contacts, schedule entries and e-mail, clipboard synchronization, printing to a PC-based printer.

Symbian 60 Phone Programming in Python

Larry Rudolph
MIT 6.883 & SMA 5508
March 2006

Installing stuff

- a package or installation file in symbian:
  - application_name.sis
- Get it onto the phone
  - push via bluetooth (or send message)
- Open up message and install
  - if there is flash memory, install it there

S60 2nd Edition, Feature Pack 2 S60_2nd_fp2_msb.zip PythonForSeries60_1_2_for_2ndEd_FP2_SDK.zip	6630, 6680, 6681	
S60 2nd Edition, Feature Pack 1 S60_sdk_2_1_NET.zip PythonForSeries60_1_2_for_2ndEd_FP1_SDK.zip	7610, 6620, 3230	
S60 2nd Edition S60_sdk_v2_0.zip	6600, 3650	

Hello World

```
import appuifw
appuifw.note(u'Hello World',u'info')
```

```
import appuifw  #  the application user interface fw(?)
u'Hello World',  #  u' '  for a unicode string.  All GUI
                 #  strings are unicode.  Others can be
u'info'          #  specifies the type of note
```

Get User Input

```
import appuifw
planet = appuifw.query(u'Which planet?',u'text')
appuifw.note(u'Hello '+planet , u'info')
```

"query()" pops up a dialog with prompt string and input type
other types can be 'number' 'date' 'code'
the '+' concatenates the two unicode strings

Get More User Input

```
import appuifw
planets = [ u'Mars', u'Earth', u'Venus' ]
prompt = u'Enter your home planet'
index = appuifw.menu(planets, prompt)
appuifw.note(u'Hello '+planets[index] , u'info')
```

The 'menu' method pops up the list of items in first param
It returns with an index into the list of menu items
Note that the prompt param must also be a unicode string

Our own interface:

sma.py

There are a bunch of annoyances in the current UI

Let's put wrappers around basic calls

We should go back and do this for location

Hiding the unicode

```
# this is file sma.py
# wrappers to appuifw, e32, and bluetooth

def note( str , type = ‘info’):
   appuifw.note( unicode(str),  type )

def query( str , type = ‘text’ ):
   return appuifw.query( unicode(str),  type )

def menu( list, prompt = ‘select one’ ):
   ulist = [ unicode(u) : for u in list ]
   return appuifw.menu( ulist , unicode( prompt ) )
```

Using sma.py

```
import sma

planets = [ ’Mars’, ’Earth’, ’Venus’ ]
prompt = ’Enter your home planet’
index = sma.menu(planets, prompt)
sma.note(’Hello ’+planets[index] )
```

No more slides

- It is easier to go through the python reference document, rather than reproducing it all here...

CSAIL