Symbian OS

a new epoc(h)

by Cédric Pulrulczyk
cedric@pulrulczyk.fr

Plan

1. Overview
2. Architecture
3. Development
4. Stack & Heap
5. Active objects
6. Client / Server Architecture
7. Security in Symbian 9

1. Overview

History

Symbian was a joint venture formed in June 1998 owned by Psion (28%), Ericsson, Motorola, Nokia, Matsushita. Psion started looking at licensing in July 1996 with the formation of Psion Software. Markets the Epoc32 operating system as the basis for mobile communication systems and smart phones. London headquaters, offices in Japan and San Francisco Bay area.

First Symbian devices : Psion Series 5mx, Psion Revo, Psion Netbook/Series 7

Evolution

Symbian OS6.1
UI: Series 60v1
Nokia 7650

Symbian OS6
UI: Series 80
Nokia 9200

Epoc R5
Ericsson R380

Symbian OS7,0
UI: UIQ
Sony Ericsson P800

Fujitsu F2102V / Siemens SX1 / Sendo X

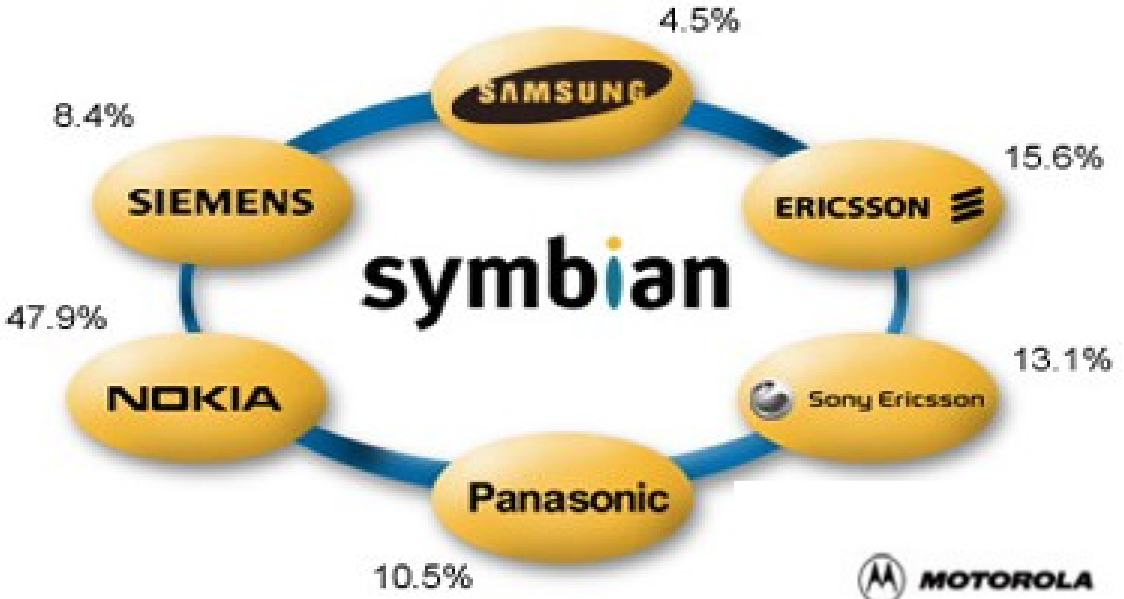
Symbian OS8.0

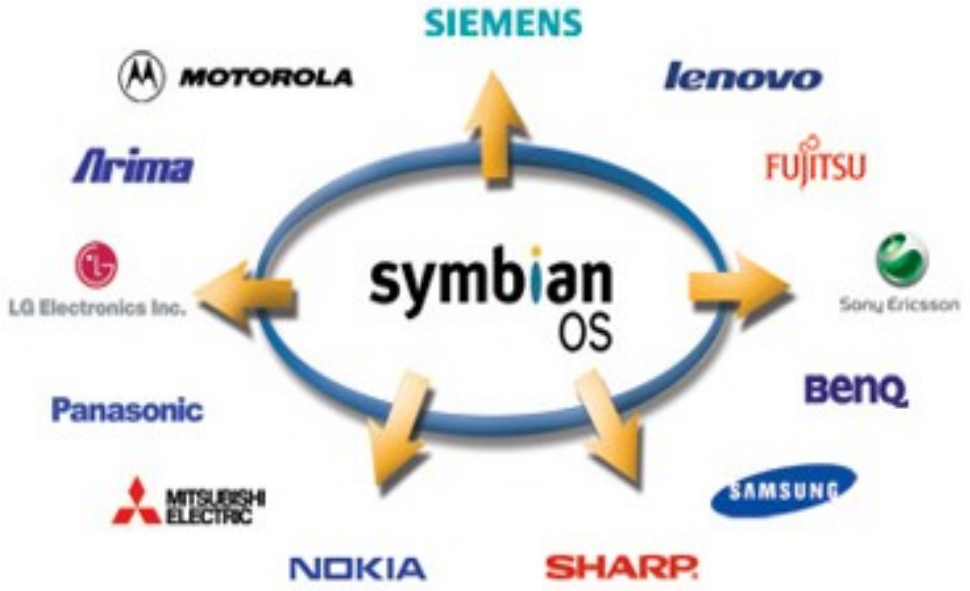
Symbian OS9.0

Lenovo P930, Motorola A1010, Panasonic X800, Samsung SGH-D720, Sendo X2, Nokia 6680, Nokia 6681, FOMA F901iC, Nokia N70, Nokia 6682, FOMA D901i, FOMA F901iS, D901iS

2000 2001 2002 2003 2004 2005 2006

Symbian today





2. Architecture

Symbian OS design

- Micro kernel design
- Battery powered
- ROM-based solutions
- Component based
- Fully object oriented
- Reliability : no loss of user data
- Simple Uis : build on GT (Generic Technology) Layers
- 95% written in C++, some C & assembler
- 32bit version
- Client-server architecture
- Class Framework
- Suite of PIM applications, email, browsers, etc...

Symbian OS



Platforms

20 %

symbian OS

Application engines
Contacts, agenda, office, SyncML, data management, browsing

Messaging
SMS, EMS, MMS, email (POP3 & IMAP4)

Java MIDP
Wireless messaging, Bluetooth, mobile media, 3D graphics (JSR184)

CLDC 1.1

20 %

Application framework
Graphical user interface framework, text and graphical utilities

Personal area networking
Bluetooth, infrared, USB

Multimedia
Images, sounds, graphics

Communication infrastructure
TCP, dual IPv4 & v6, HTTP, WAP stack

55 %

Security
Cryptography, certificate management, software installation

Telephony
GSM, GPRS, HSCSD, EDGE, CDMA (IS-95), cdma2000, WCDMA

Base
User library, file server, Kernel, device driver

5 %

Component architecture

Applications

SHELL | AGENDA | WORD | RECORD | WORLD | DRAW | COMMS | BROWSER

OPL | DATA | SHEET | TIME | RECORD | SPELL | MAIL | NETWORK

System

DIALOGS | MENU | TOOLBAR | ICONS | RESSOURCES | JAVA VM

RICH TEXT LAYOUT | GRID CONTROL | EDIT CONTROL | LIST CONTROL | APP FRAMEWORK | JAVA CLASS LIBRARIES

Servers

WINDOW SERVER | FONTBITMAP SERVER | PROCESS SERVER | SOCKET SERVER | SOUND SERVER | WIRELESS SERVER

DATABASE SERVER | FILE SERVER | WORLD SERVER | ALARM SERVER | COMMS SERVER | WIRELESS PROTOCOL

BASE

EUSER.DLL

EXECUTIVE

KERNEL.DLL

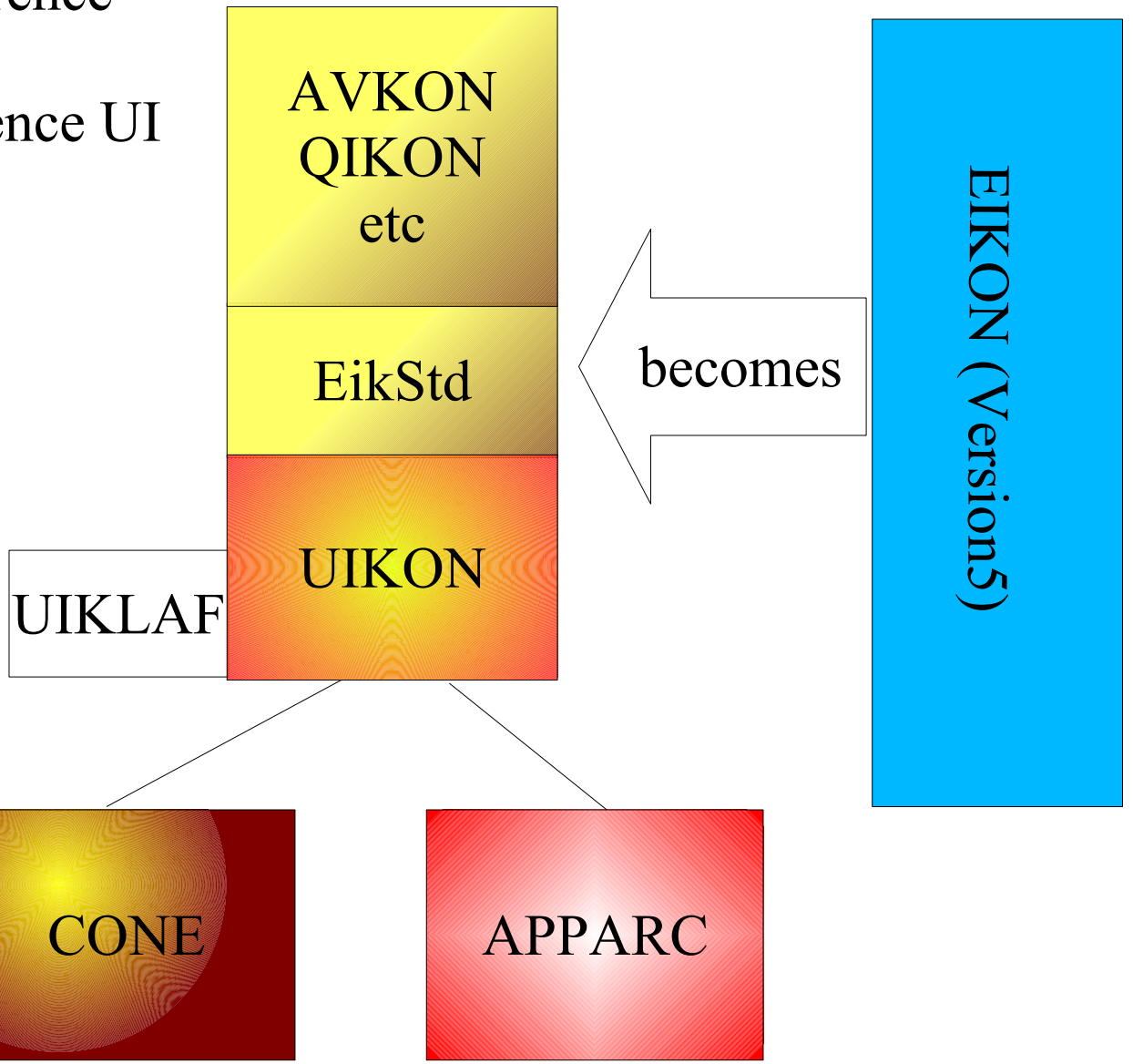
HAL

SUPERVISOR SERVER

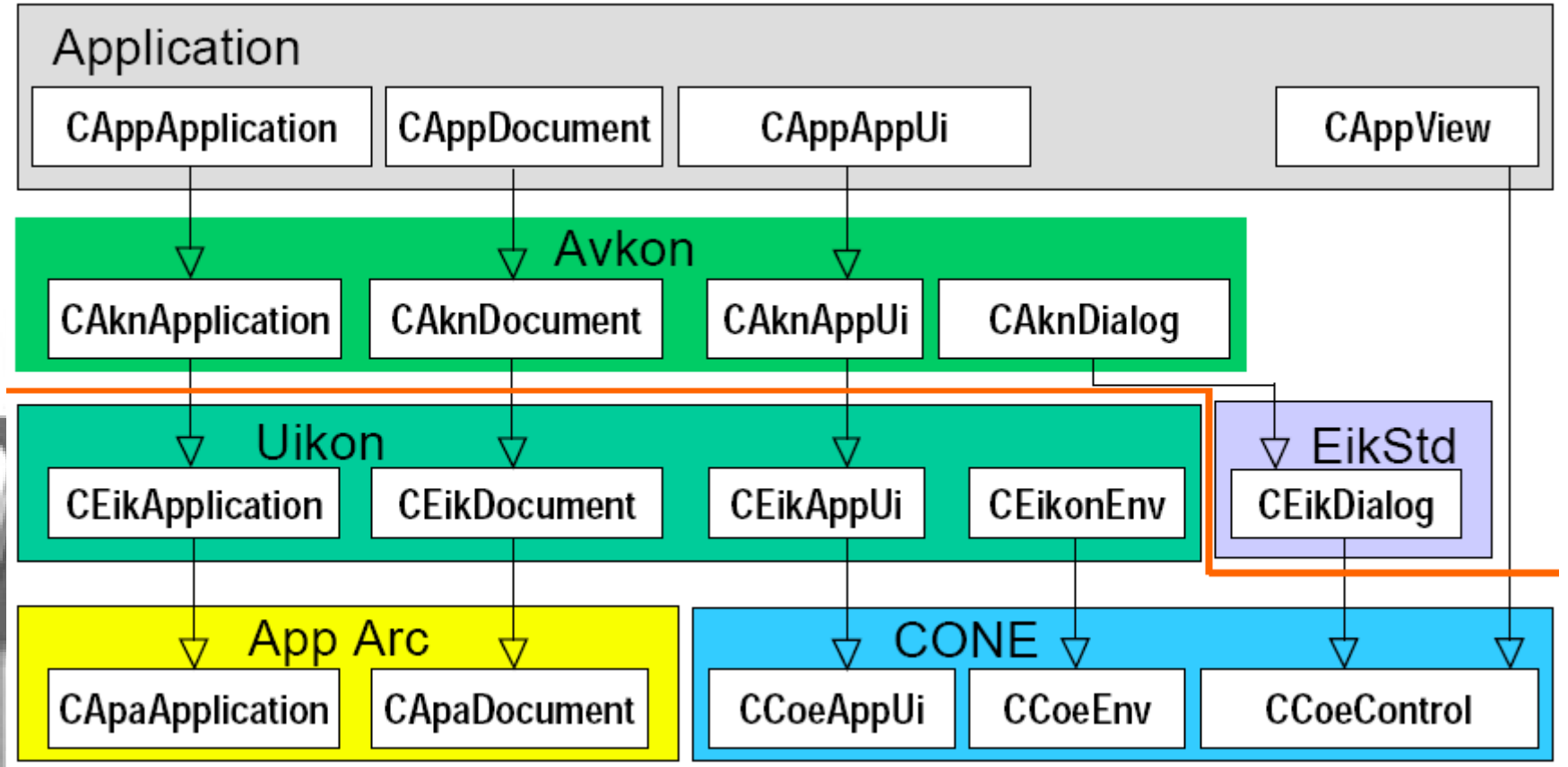
SOUND DRIVER | SERIAL DRIVER | USB DRIVER | TIMER DRIVER | KEYBOARD DRIVER | POINTER DRIVER | PARALELL DRIVER | DIGITIZER DRIVER

UI and graphics

- Avkon is the reference UI for Series 60
- Qikon is the reference UI for UIQ



Application framework – Symbian & S60



3. Development

C++ Development products available

symbian

Symbian OS - the mobile operating system

DevKit

Product Technology developers

CustKit

UI Vendor

SDKs

After-market application vendor

Choosing a C++ Development product

Considerations:

- where to hit the device development lifecycle?
- which APIs are needed?
- which devices are targeted?

SDKs

- add-on application and some system level functionality
- UI platform specific code
- being developed after ‘lead device’ ships

Devkits

- lower level code -device driver, filing system, base port
- intense system plug-in development -e.g. MMF plug-in
- generally applicable to any Symbian OS product
- aiming to ship in ROM, as a component of products
- when developing along with hardware components

C++ Build Tools & IDEs

- Primarily command line build tools used to set up / configure projects

- Then use an IDE for development

- Currently supported C++ IDEs include

... Nokia Carbide

…Borland C++ Mobile Edition

…Microsoft Visual C++ can be used on some versions

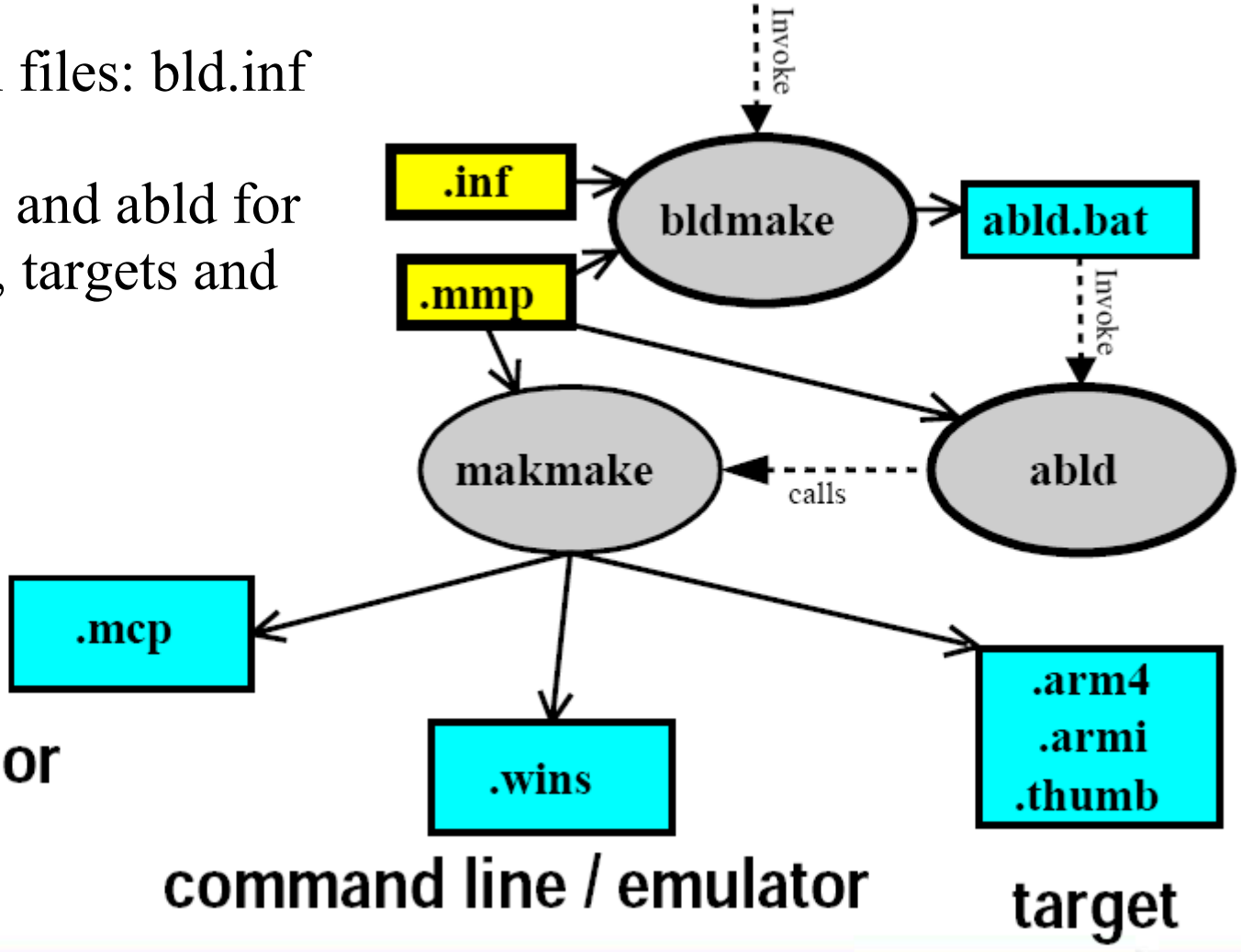
Carbide Development Tools

Borland

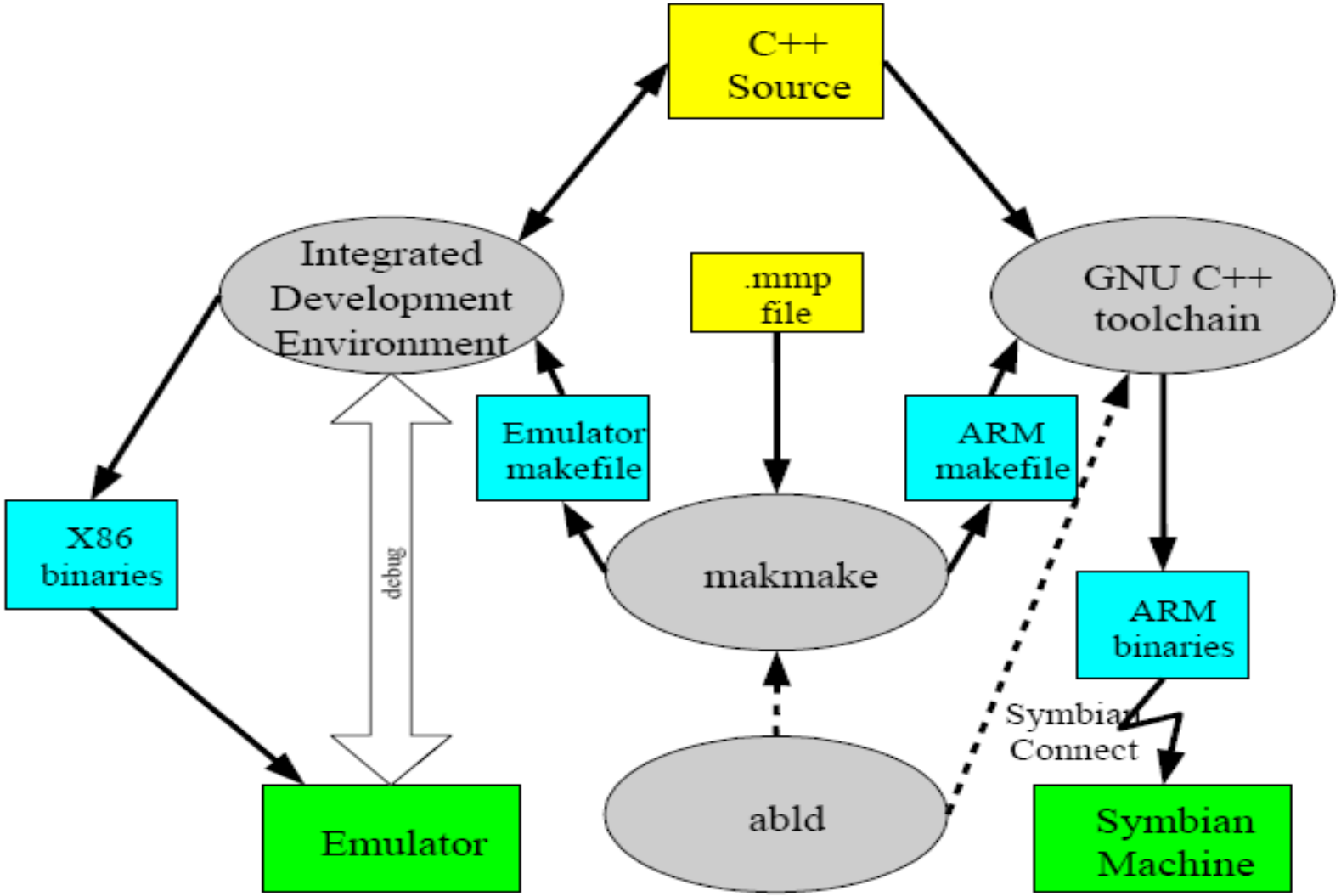
Microsoft

Bldmake & abld commands

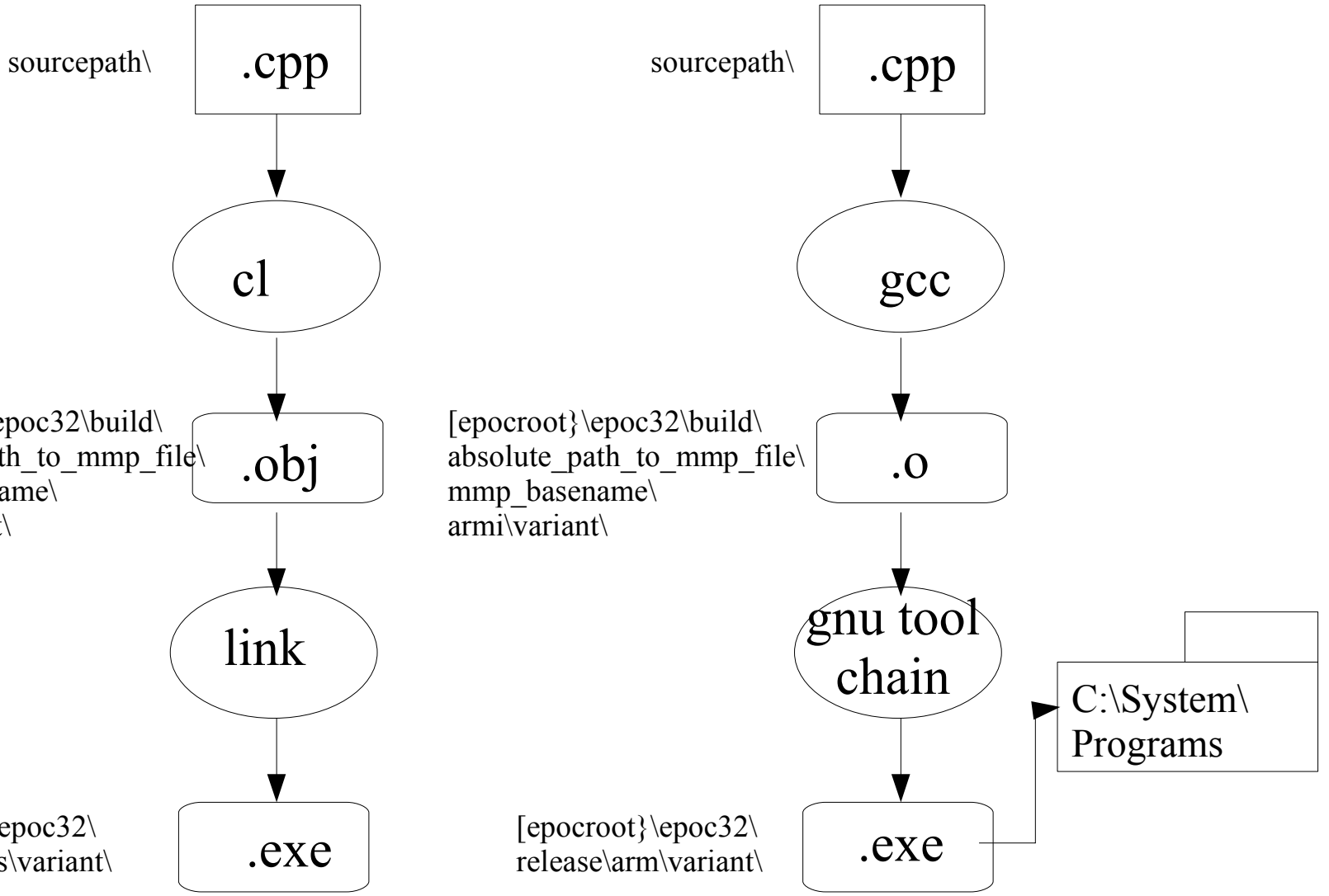
Specification files: bld.inf and .mmp
Use bldmake and abld for all platforms, targets and variants



Cross Building C++ for Emulator/Target

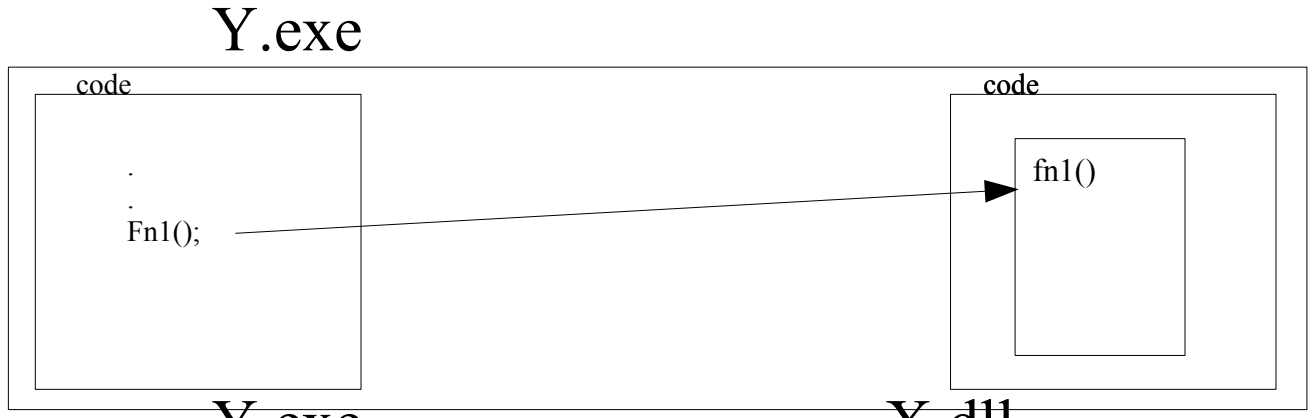


Building a .exe

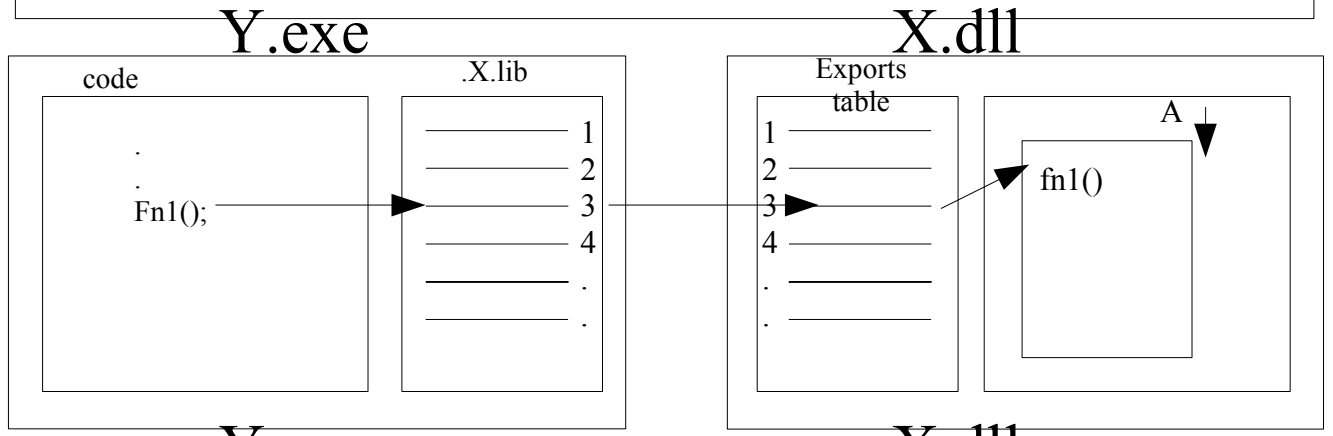


DDL or EXE

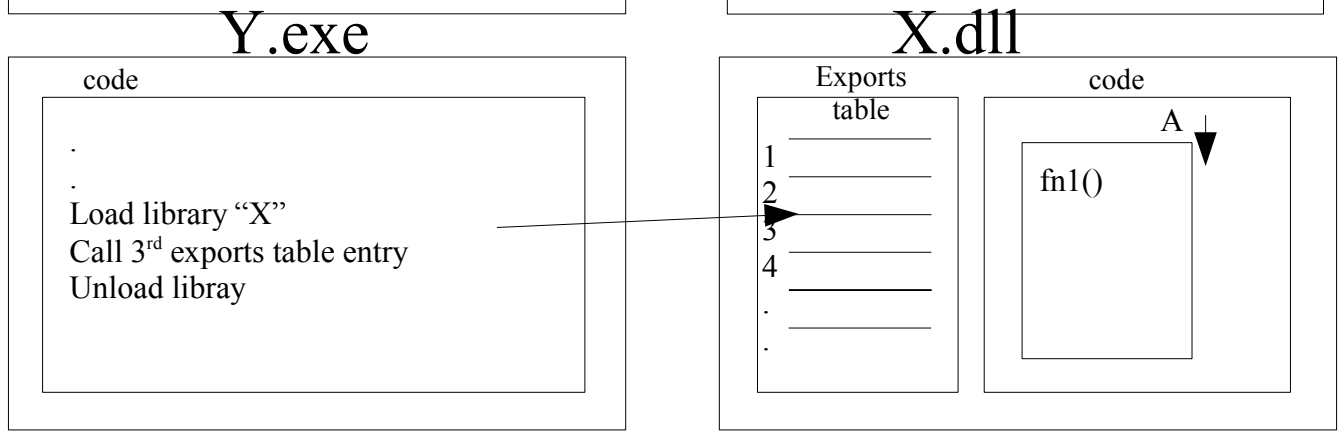
•Exe only
- Link and load exe
- Static resolution of call



•Exe and shared library
- Link and load both
- Auto by system
- Indirection via. lib table
- Static resolution of cail



•Exe and provider library
- Link and load exe
- Dynarnic load of dll
- Dynamic resolution of call



EXEs versus DLLs

EXE	**DLL**
Server or command line application	Library of arbitrary code
Single entry point	Multiple entry points
Code is stand-alone, cannot be shared	Code is shared by multiple clients
All in memory when the .exe runs	Can be unloaded when no clients
	Can be shared library or provider library

UIDs

	UID1	UID2	UID3
Helloworld.exe	EXE	0	0
Engine.dll	DLL	DLL	Unique to this DLL
Helloworld.app	DLL	APP	Unique to this application
Hellodata1	DOC (direct or permanet)	DOC	
Helloworld.aif	DOC (direct)	AIF	

Example .mmp for .dll

```
TARGET          CreateStaticDLL.dll
TARGETYPE       dll
UID             0x1000008d 0x10004268
SOURCEPATH      .
SOURCE          CreateStaticDLL.cpp
USERINCLUDE     .

SYSTEMINCLUDE   \Epoc32\include
LIBRARY         euser.lib

EXPORTUNFROZEN
```

Symbian C++ Coding Conventions/C++ Concepts

Why

…help reinforce Symbian OS programming idioms
…provide common syntax for Symbian OS C++
…help communicate Symbian OS C++ class design for APIs
…serve as checklist for coding
…act as reference during code inspection/review

Concepts

- Naming conventions
- Error handling and cleanup

…C++ exceptions not used

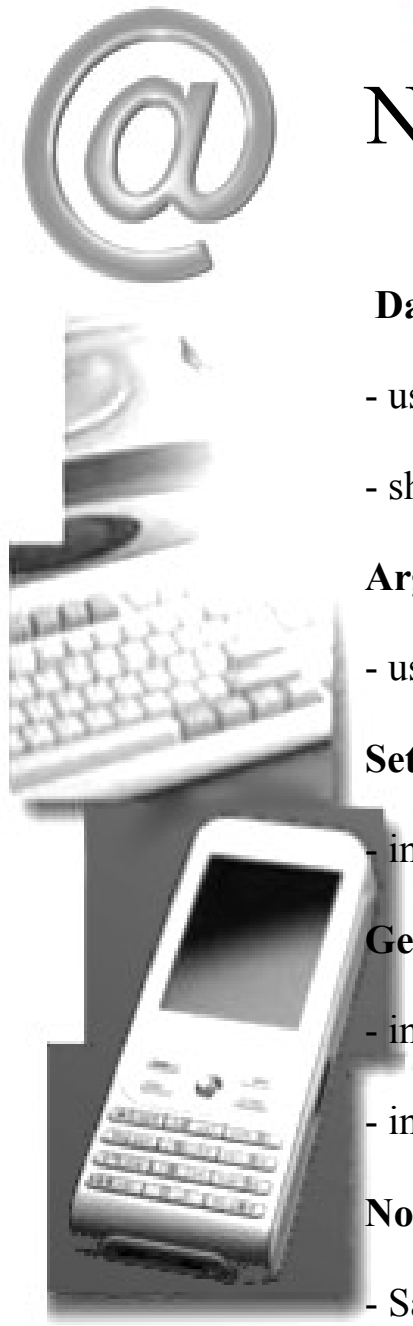
- Descriptors (length checked strings)
- Active objects

…multitasking and event handling

- Porting generic C or C++ code

Naming conventions

- **T Types**
  …automatic variables, e.g. TInt
- **C Classes**
  …heap allocated, requiring cleanup
- **M Classes**
  …mixins for callbacks, like Java interfaces
- **R Classes**
  …handle to remote resource, require cleanup
- **Static Classes**
  …e.g. User



Naming conventions

Data members

- use prefix i ... (e.g. i Count)

- should be private

Arguments

- use prefix a... (e.g. aParam)

Setter functions

- inline void SetThing(const TType& aThing)

Getter functions

- inline const TType& Thingo const

- inline void GetThing(TType& MyThing)

Note getters/setters should be explicitly declared as infine

- Same applies to all trivial functions

ASSERT macros

•Assert any condition which must be true in order for your code to execute correctly

•__ASSERT_ALWAYS

- release and debug builds
- will break production code

•__ASSERT-DEBUG

- debug builds only
- extra checking to find bugs early

• Syntax:

- __ASSERT-ALWAYS (c, p) / -ASSERT-DEBUG (c, p)

• Defined as:

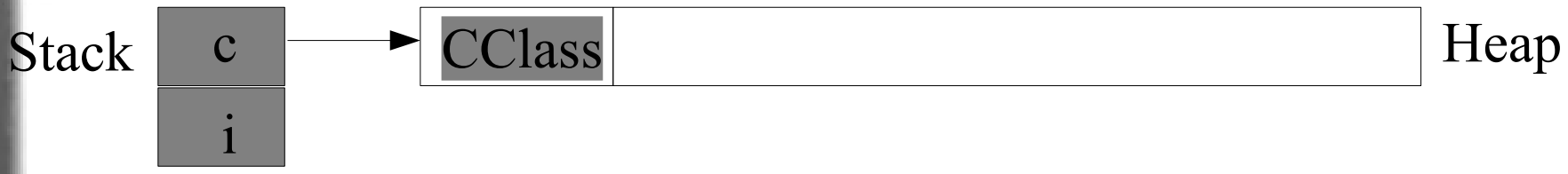
- Test conditional expression c
- If false, call function p (usually a panic)

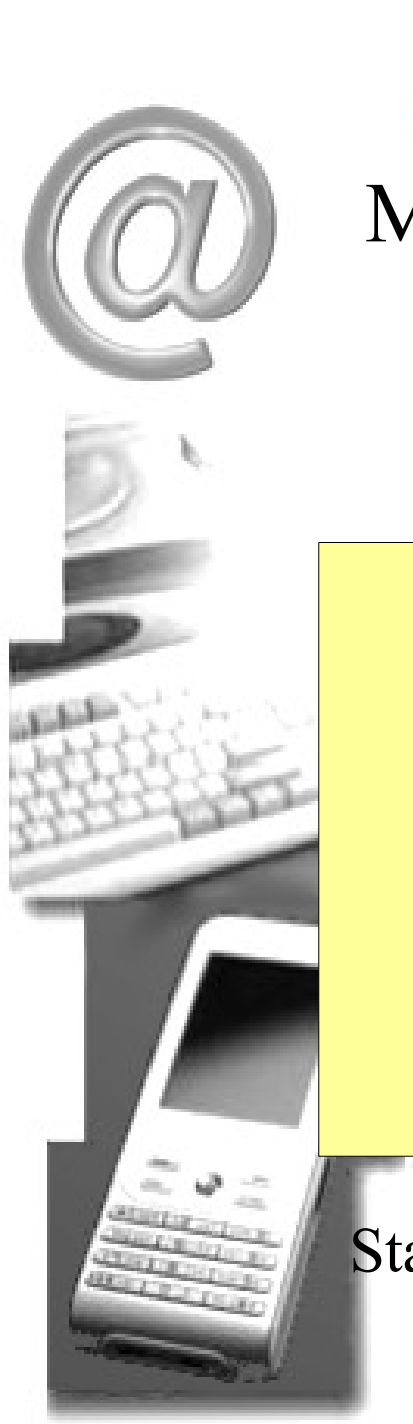
4. Stack & Heap

Stack & Heap Objects

- Stack: object deletion automatic
- Heap: object deletion explicit by programmer
- Pointers on stack point to heap

```
{
TInt i;
CClass* c = new(ELeave) CClass;
...
```





Memory Leak

Can only free heap memory if have a pointer to it

```
{
TInt i;
CClass* c = new(ELeave) CClass;
c->Use();
// delete c;
}
```

Leaving on Exception

```
CClass* CClass::NewL(TInt aInt,CBase& aObj)
{
CClass* self=new(ELeave)CClass(aInt);
CleanupStack::PushL(self);
self->ConstructL(aObj);
CleanupStack::Pop(self);
return self;
}
```

Leaving on Exception

```
TInt err;
TRAP (err,CreateObjectL());
```

```
void CreateObjectL()
{
CObject* obj=new (ELeave)CObject;
```

- Write code assuming success
  - …if exception occurs, then Leave -call User::Leave()
- Run it within a harness that traps exceptions

TRAP Harness Macros

```
TRAPD(err,LoadFileL());
if (err!=KErrNoMemory)
```

- If exception occurs leave, returning to last harness
- Two Macros: `TRAP`& `TRAPD`
  - …`TRAPD` Declares `err` as `Tint` and `=KErrNone`
- Functions that leave end in a `L`
- Design where TRAPs needed from start
- Expensive, use sparingly - invariably high up

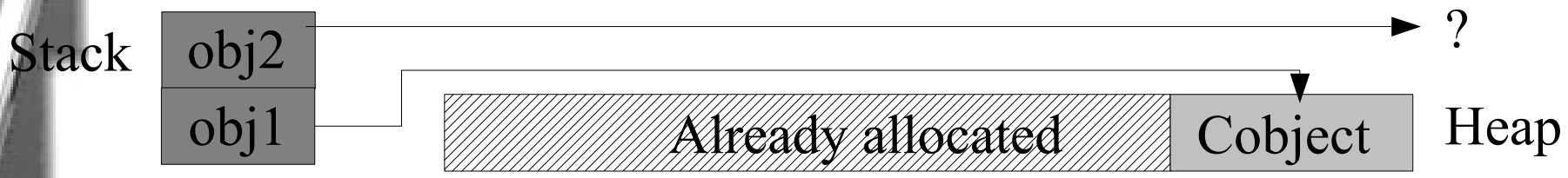
Cleanup Stack

```
CClass* CClass::NewL(TInt aInt,CBase& aObj)
{
CClass* self=new(ELeave) CClass(aInt);
CleanupStack::PushL(self);
self->ConstructL(aObj);
CleanupStack::Pop(self);
return self;
}
```

Cleanup Stack

```
TRAPD (err,CreateObjectsL());
  ...
  void CreateObjectsL()
  {
  CObject* obj1=new(ELeave)Cobject;
  CleanupStack::PushL(obj1);
  CObject* obj2=new(ELeave)CObject;
```

If second allocate fails, we have a pointer to obj1



Descriptors

- Descriptors are fundamental classes
- They handle strings & binary data (unlike ‘C’ strings which mustbe null terminated)
- T-Descriptors behave like built-in types e.g `TInt`, can be safely created and orphaned on stack
- All descriptors carry length and maximum length information so:

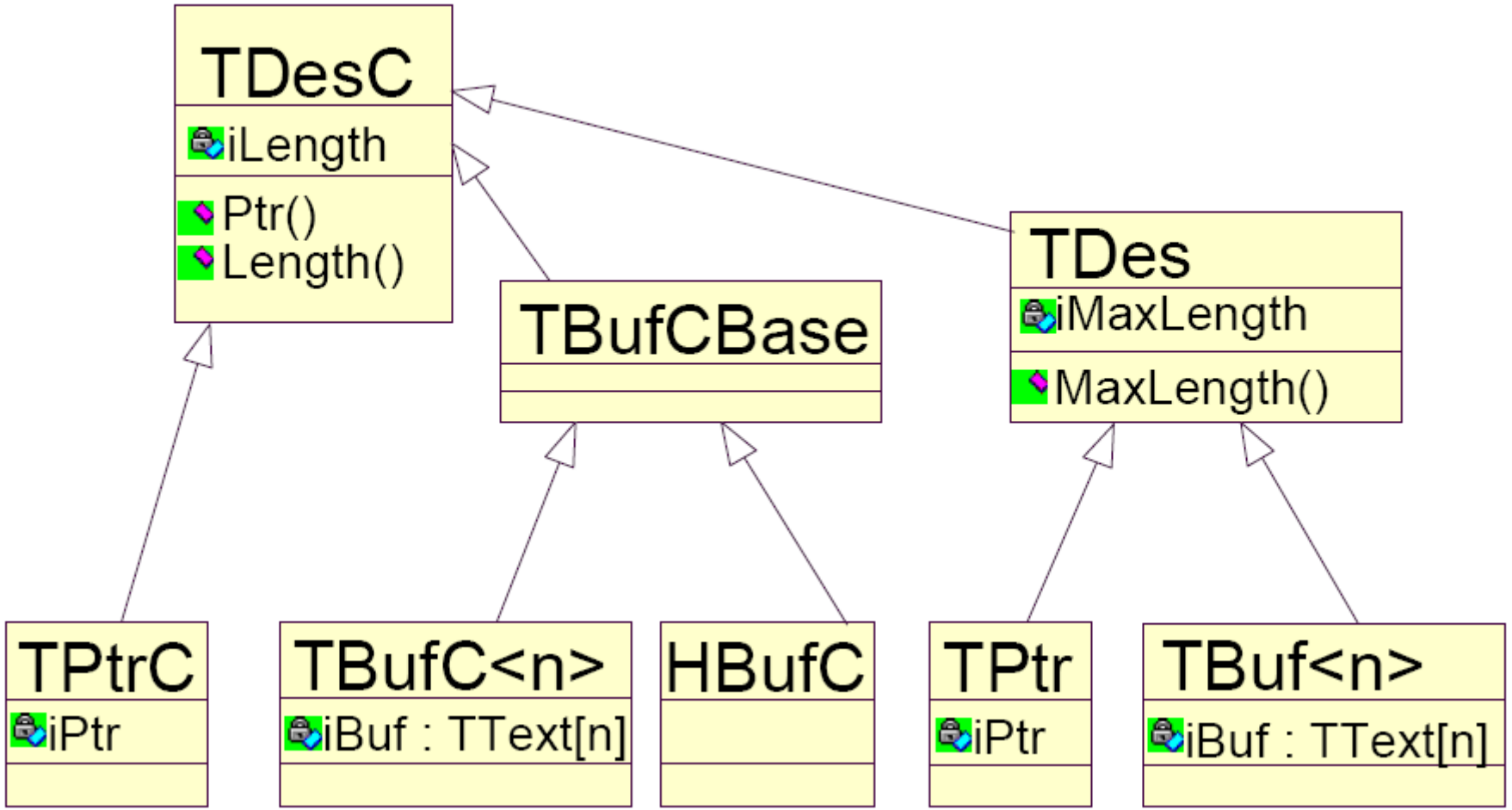
  …impossible to construct a descriptor with a length which exceeds the buffer’s capacity (panic)

  …impossible to assign data into a descriptor with a length which exceeds the buffer’s capacity (panic)

  …modifiable descriptors:TBuf,TPtr

  - impossible to alter the length of an existing descriptor beyondiMaxLength(panic)

Descriptor classes



Typical Descriptor Usage:

- TBuf/TBufC: small amounts of storage on the stack, or whenever the maximum size is known at compile time
- TPtrC: referencing constant strings or data
- TPtr: referencing strings or data, modifying non-modifiable buffers via Des()
- HBufC: larger amount (or initially unknown amount) of data

5. Active objects

Active objects -Background

- Symbian OS is a heavily asynchronous system
  …applications spend most of time waiting for UI events and system services
  …possible multi-tasking implementation is shared memory/multi-threading
  …but most Symbian OS applications use an Active Object framework
- What is an Active Object framework?
  …a paradigm for non-pre-emptive multi-taskingwithin a single-threaded application

Active Object Classes

- Active Object: derived from **Cactive** class. Encapsulates:
  …asynchronous request (e.g. to server)
  …application's handler for the completed request
  …plus functionality for cancellation of the request!

- Active Scheduler: derived from **CactiveScheduler** class.
  …schedules Active Objects non-pre-emptively within an application (AOs have priorities but cannot pre-empt each other)
  …encapsulates waitloop processing of events
  …one per thread

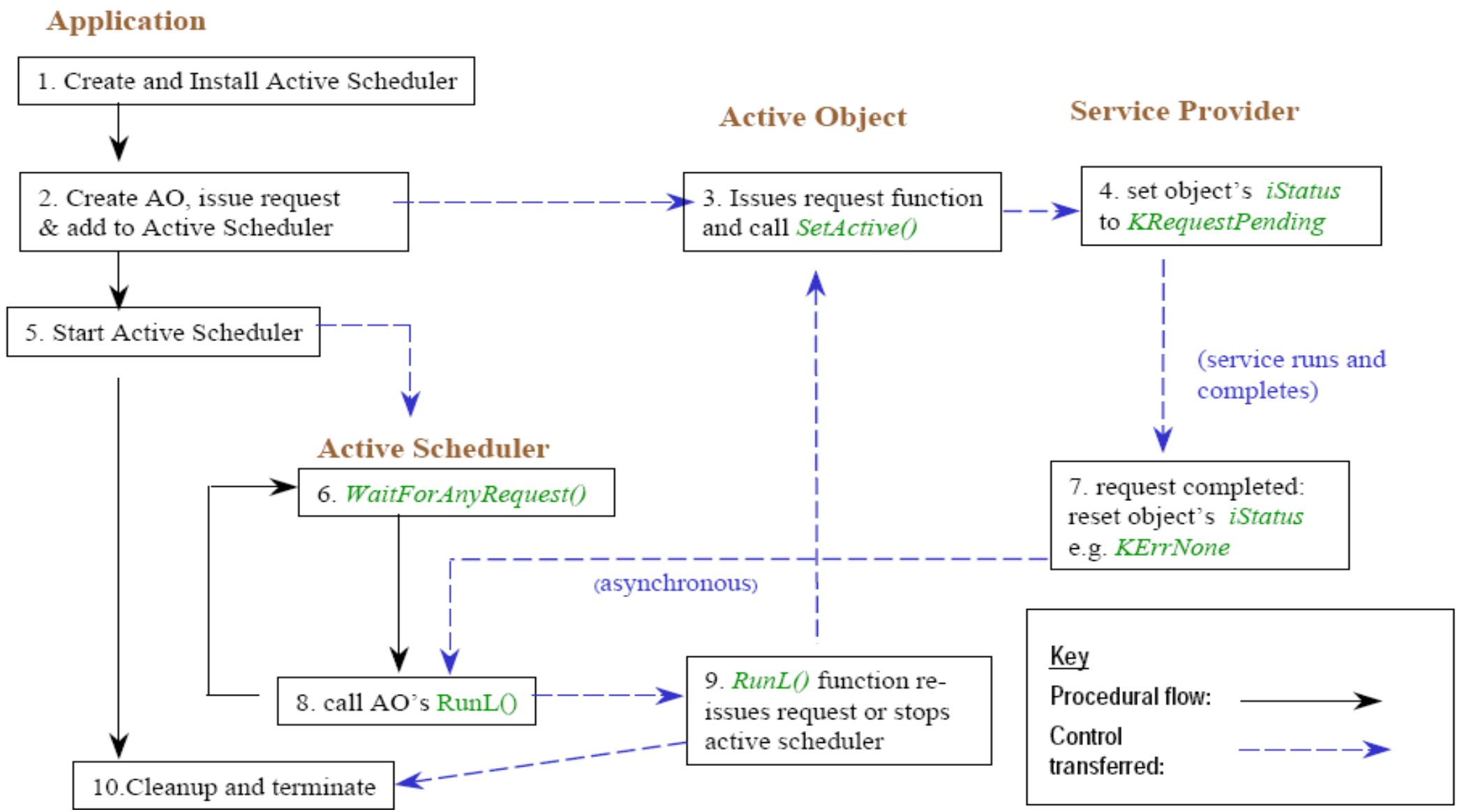
CActive class basics

```
class CActive : public CBase
{
public:
IMPORT_C void Cancel(); // Cancel outstanding wait
IMPORT_C void Deque();
IMPORT_C void SetPriority(TIntaPriority);
inline TIntPriority() const;
protected:
CActive(TIntaPriority);
void SetActive();
virtual void DoCancel() =0; // Implement the Cancel protocol
virtual void RunL() =0; // Handle Event
virtual TInt RunError(TInt aError);
private:
TBool iActive;
public:
TRequestStatus iStatus;
};
```

CActiveScheduler class Basics

```
class CActiveScheduler : public CBase
{
public:
IMPORT_C CActiveScheduler();
IMPORT_C ~CActiveScheduler();
IMPORT_C static void Install(CActiveScheduler*
  aScheduler);
IMPORT_C static void Add(CActive* anActive);
IMPORT_C static void Start();
IMPORT_C static void Stop();
IMPORT_C virtual void WaitForAnyRequest();
IMPORT_C virtual void Error(TInt anError) const;
};
```

Life cycle of AO



6. Client/Server Architecture

System servers

Kernel Server

File Server

Serial Comms Server

Sockets Server

Telephony Server

Window Server

Font & Bitmap Server

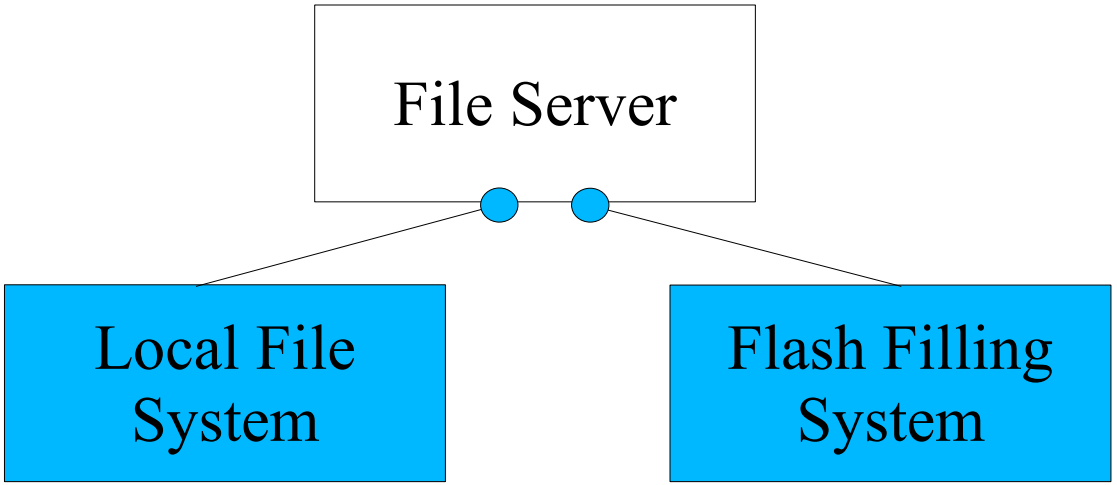
Media Server

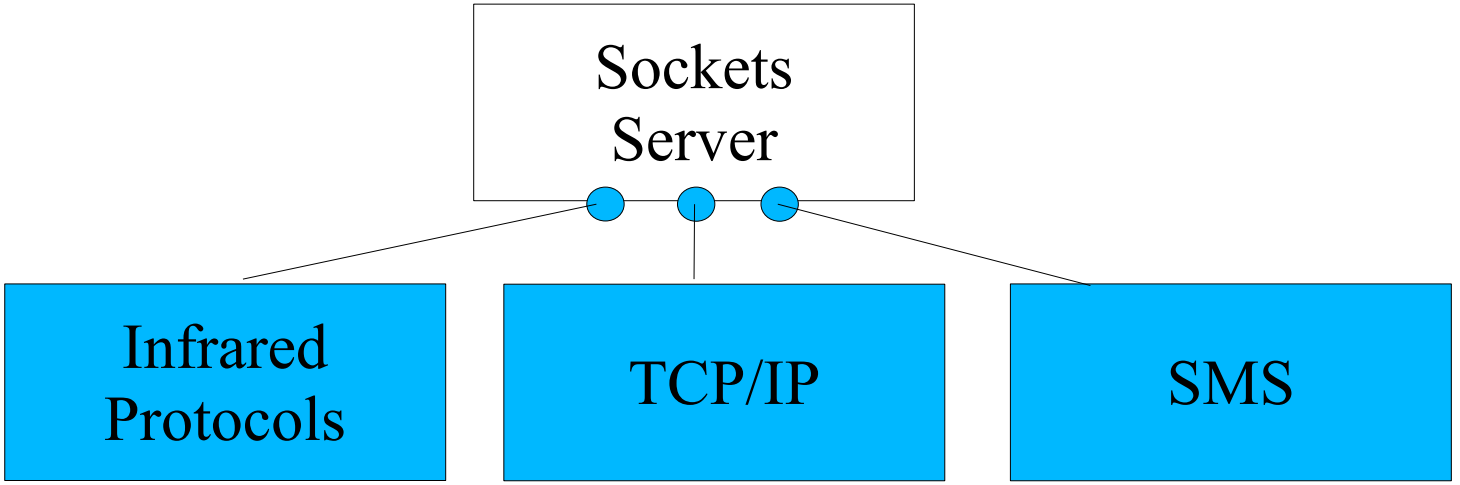
Eikon Server

View Server

Message Server

Plug-in Architecture





Client/Server Overview

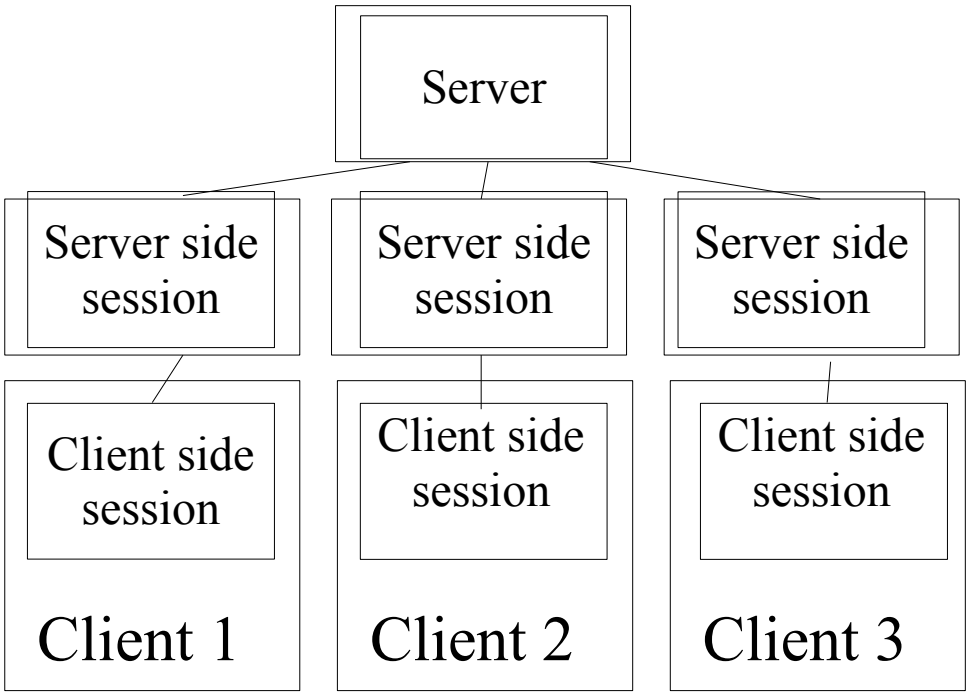
- Session-based
  - multiple sessions
  - Subsessions
- Client Service API
- Inter-thread communication
- Safe
  - isolation in separate process
  - message passing between client & server

Using a simple server

```
RCountServSession session;
TRequestStatus status;

User::LeaveIfError(session.connect());

_LIT(KTxtLegalString,"224");
ret = session. SetFromString(KTxtLegalString);

session.DelayedIncrease(l, 1000000, status);
User::WaitForRequest(status);

session.DecreaseBy(3);

session.close();
```

Multiple clients

•Example system with 1 server and 3 client threads

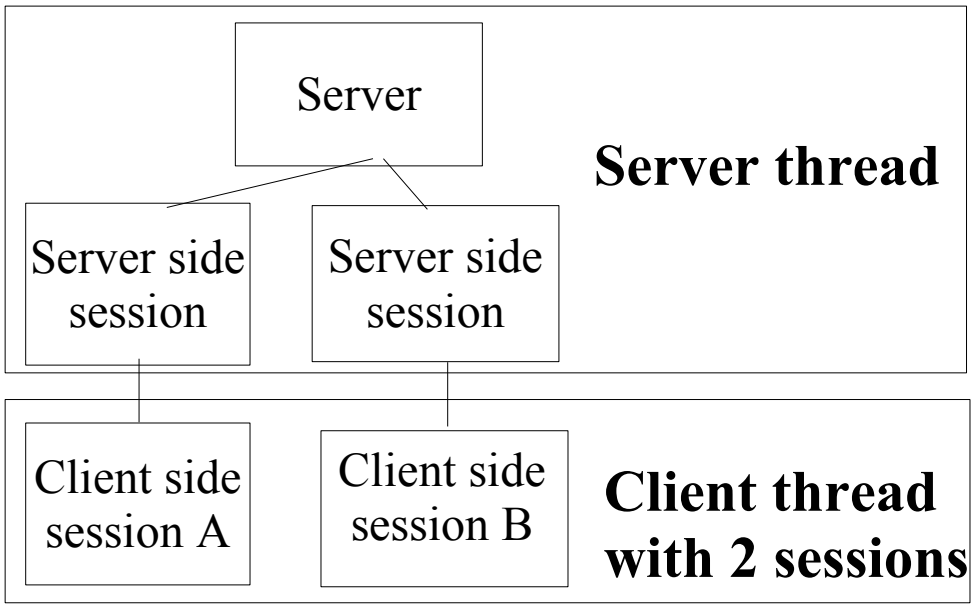


•Server-side:
- 1 server
- 3 sessions

•Client-side
- 3 sessions
- each in its own thread

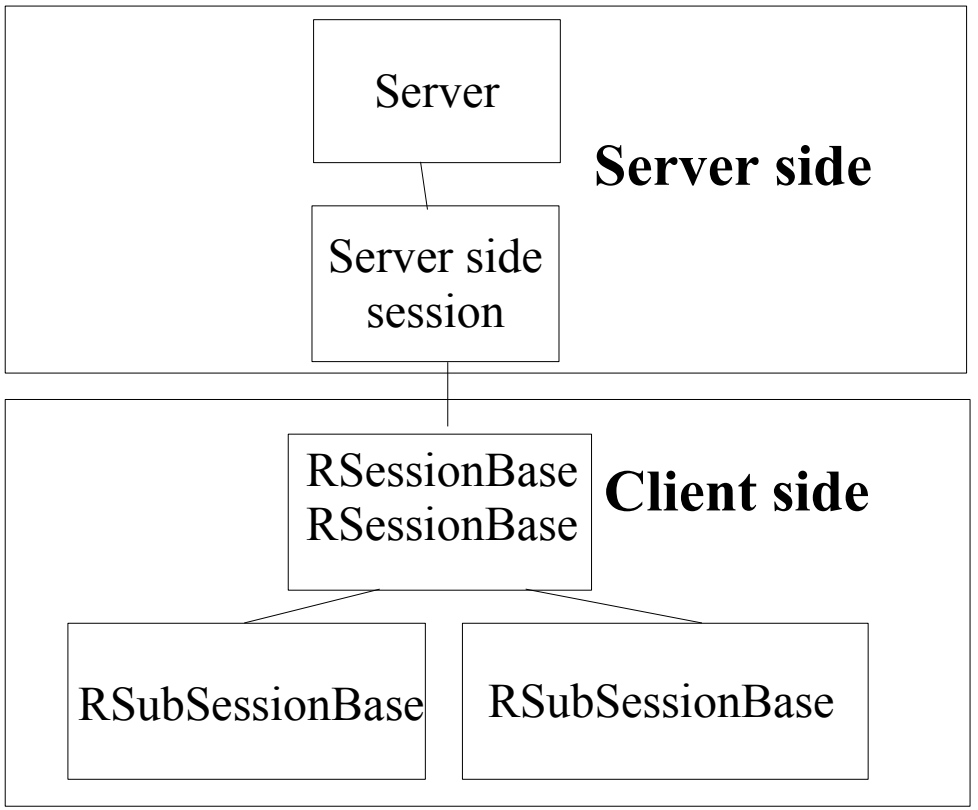
Multiple sessions

- Example system with 1 server and 1 client thread
- 2 client sessions exist within a single thread



Subsessions

Better (more lightweight) way for a single client to have multiple logical connections with one server.



Client-side classes (1)

• Two classes are used to produce client side APIs...
- RSessionBase
- RSubSessionBase

• Derived classes must:
- Create sessions or subsessions as appropriate
- Send request messages from the client to the server

Client-side classes (2)

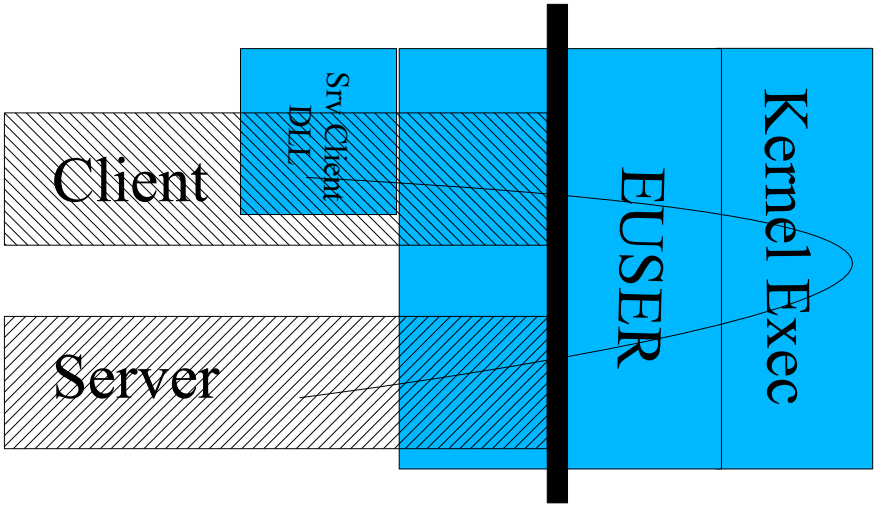
- Create a session with
  - RSession Base:: CreateSession()
- Create a subsession with
  - RSubSessionBase:: CreateSubSession()
- Send messages with
  - Send(TInt aFunction,TAny* aPtr);
  - SendReceive(Tlnt aFunction,TAny* aPtr,TRequestStatus& aStatus);
  - SendReceive(Tlnt aFunction,TAny* aPtr);

Client-side DLL

- Client Side API often built as DLL
- Access services using export library

File Server: efsrv.lib

Socket Server: esock.lib



Messages (1)

- Clients send messages to servers
- Each message is a request for a service
  - Server processes request
  - Performs work on behalf of client
  - (Optionally) sends a reply to the client
- Two types of message:
  - synchronous
  - asynchronous

Messages (2)

- Request messages simply consist of five 32-bit numbers
  - A function number (or 'opcode')
  - Four 32-bit parameters

- Each of the 32-bit parameters can be:
  -An integer, or
  -A pointer to a flat data structure, wrapped as a descriptor

A real API

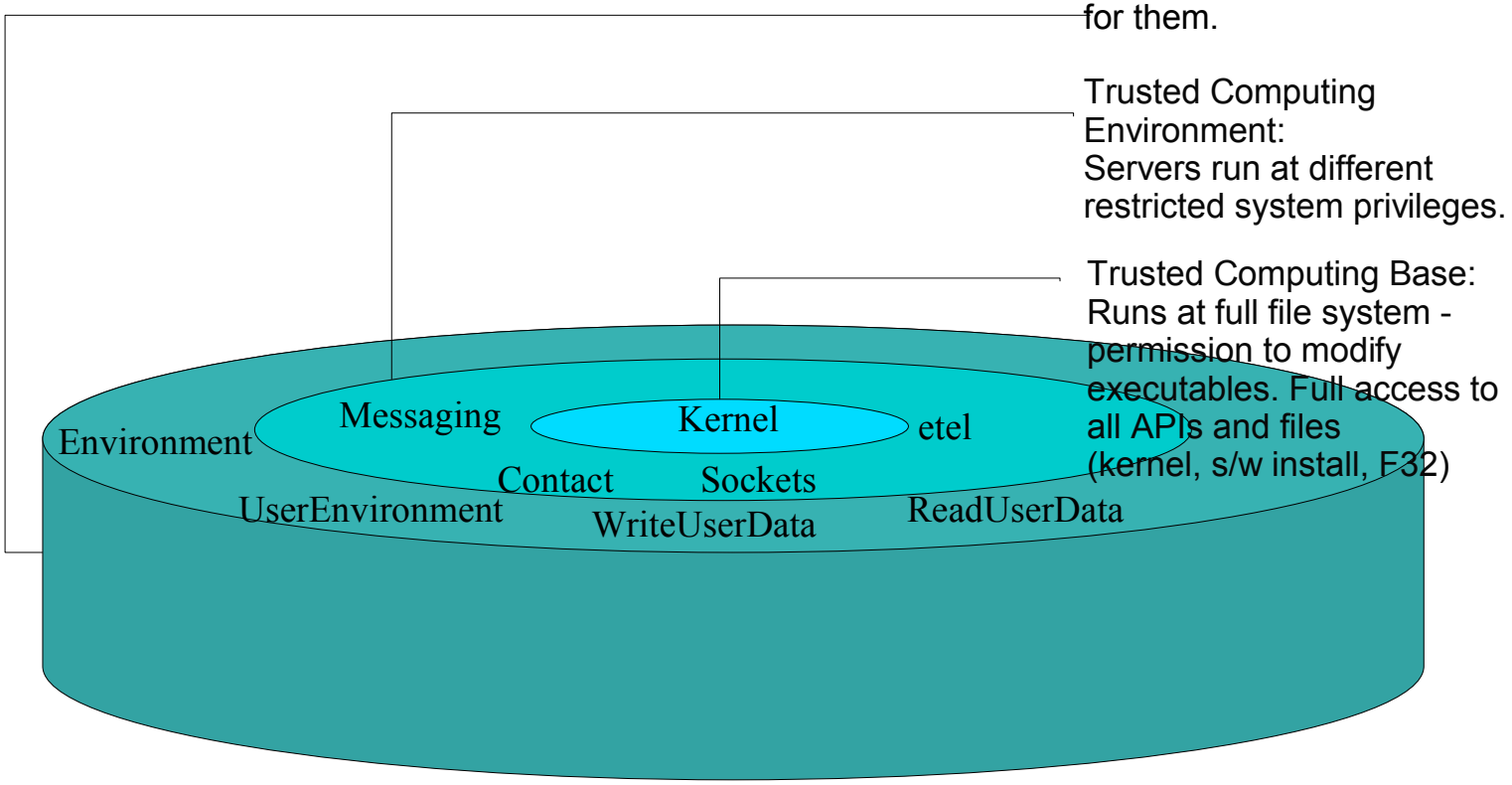
```
Class RFs : public RessionBase {...}
Class RFsBase : public RSubSessionBase {...}
Class RFile : public RFsBase
{

public:

IMPORT_C TInt Open(RFs& aFS, const Tdesc&  aName, TUint aFileMode);
IMPORT_C TInt Read(TDes8& aDes) const;
IMPORT_C void Read(TDes8& aDes,
      TRequestStatus& aStatus) const
IMPORT_C TInt Write(conSt TDesC8& aDes);
IMPORT_C void Write(conSt TDeSC8& aDes,
TRequestStatus& aStatus);
};
```

Example :

```
void CClass::SomeFunctionL()
{
RFs fs;
CleanupClosePushL(fs);
user::LeaveIfError(fs.connect());
user: LeaveIfError(fs.Delete(KSomeFi1eName));
CleanupStack::PopAndDestroy();
}
```

7. Security in Symbian 9

Security with Symbian 9



User can grant these capabilities at install time OR applications can be signed for them.

Trusted Computing Environment:
Servers run at different restricted system privileges.

Trusted Computing Base:
Runs at full file system - permission to modify executables. Full access to all APIs and files (kernel, s/w install, F32)

Policing capabilities

When calling a capability-protected API, verification of the capability may be dependent on the arguments passed to the API:
For example, a call to `CFileMan::Copy()`:

- **is checked** for **AllFiles** if it is accessing a protected folder (e.g. \private\<otherSID>)
- **is not checked** for any capabilities if it is accessing the application's protected data directory (in \private\<mySID>) or any public file

Therefore, most applications will probably not require AllFiles capability.

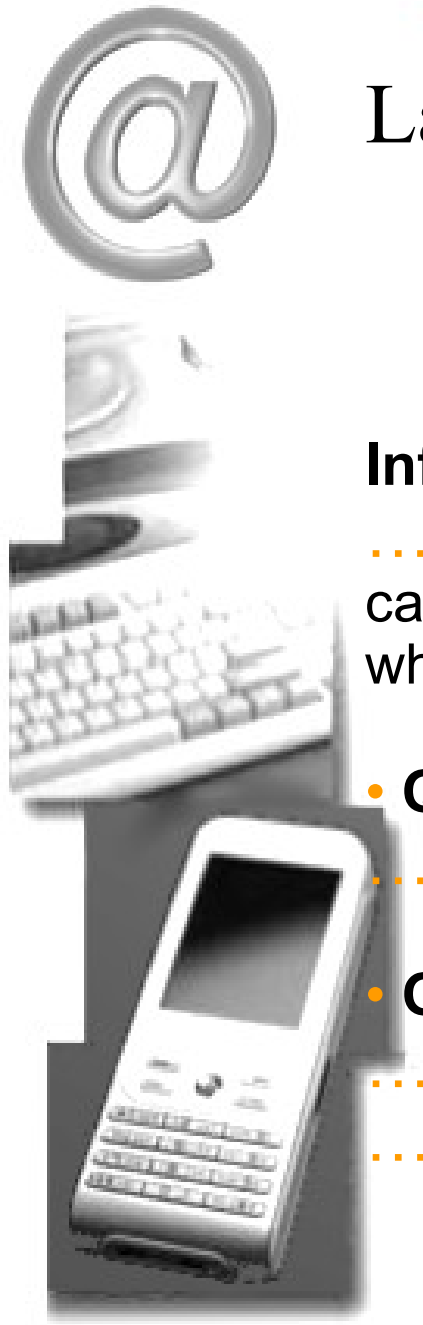
Capabilities

Rule 1: The capabilities of a process never change

- No way to add or remove capabilities to a process
- Loading a DLL never changes the process' capabilities

Rule 2: A process cannot load a DLL with less capabilities than itself

- DLL capabilities do *only* reflect a level of trust
- DLL capabilities do not authorise anything
- DLL code runs at process' capabilities level
- DLL can have more capabilities than process



Last question.

Information

…A Bluetooth game application Game.exe, with LocalServices capability, wants to load and call functions in GameEngine.dll which only computes game state and has no capabilities.

- **Question**

…Can Game.exe load GameEngine.dll?

- **Opinions**

…A - I think it can

…B - I don't think it can • The situation is "**B** – cannot"

Capability assignment - syntax

Capabilities are defined in MMP files

```
// program123.mmp
TARGET program123.exe
TARGETTYPE exe
UID 0x00000000 0x00000123
SOURCEPATH ..\mysource
SOURCE myfile.cpp
USERINCLUDE ..\include
SYSTEMINCLUDE \epoc32\include
…
CAPABILITY ReadUserData WriteUserData
```

- Applications requesting capabilities must generally be signed

Signed and Unsigned Capabilities

	One Shot	Blanket
Unsigned - Sandboxed	NetworkServices Location	LocalServices UserEnvironment
Signed	Premium Billable events. (Not capability related)	LocalServices UserEnvironment NetworkServices Location ReadUserData WriteUserData ReadDeviceData WriteDeviceData SWEvent ProtSrv PowerMgmt SurroundingsDD

Thank you for your attention