Porting applications to BeOS and Zeta

Case study: XEmacs

Introduction

Recurrent problem of the BeOS platform: lack of software.

We are getting more native applications,
but some are too complex to reimplement natively.

Considerations (1: supported platforms)

Is the app already multiplatform ?

→ Yes: there is probably a defined interface
for supporting new platform backends

→ No: How hard is it to add a second platform in the app ?

→ Is it worth it ? Maybe native app would be faster...

Considerations (2: low-level APIs)

What kind of API does the app use ?

→ POSIX: quite easy, for most stuff.

→ WIN32: Not so different from POSIX
hint cygwin implements POSIX from WIN32...
just need the other way round.

→ abstract: abstraction layer in the app: just need to write the wrappers.

→ Other: pray :-)

Considerations (3: User Interface)

Type of the app ?

- → Terminal (CLI, curses):
  smile :D

- → Minimal GUI (dialogs, alerts):
  maybe popen("/bin/alert") could work ?
  some simple C wrapper around GUI code...

- → "Flat" GUI (like Bochs, many games):
  backend API to provide the app with a virtual framebuffer,
  and basic keyboard input.

- → Toolkit based GUI:

  - → Multi-platform already ?

  - → Identify the complexity of toolkit usage

  - → Disable complex controls ?

Considerations (4: Thread safety)

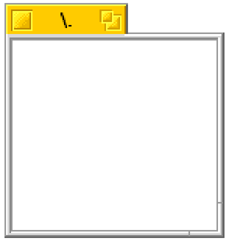
→ App uses threads ?

→ Is the work code rentrant ?

If yes it might be called from a BLooper object regularily.

If not we need to serialized the input the app needs.

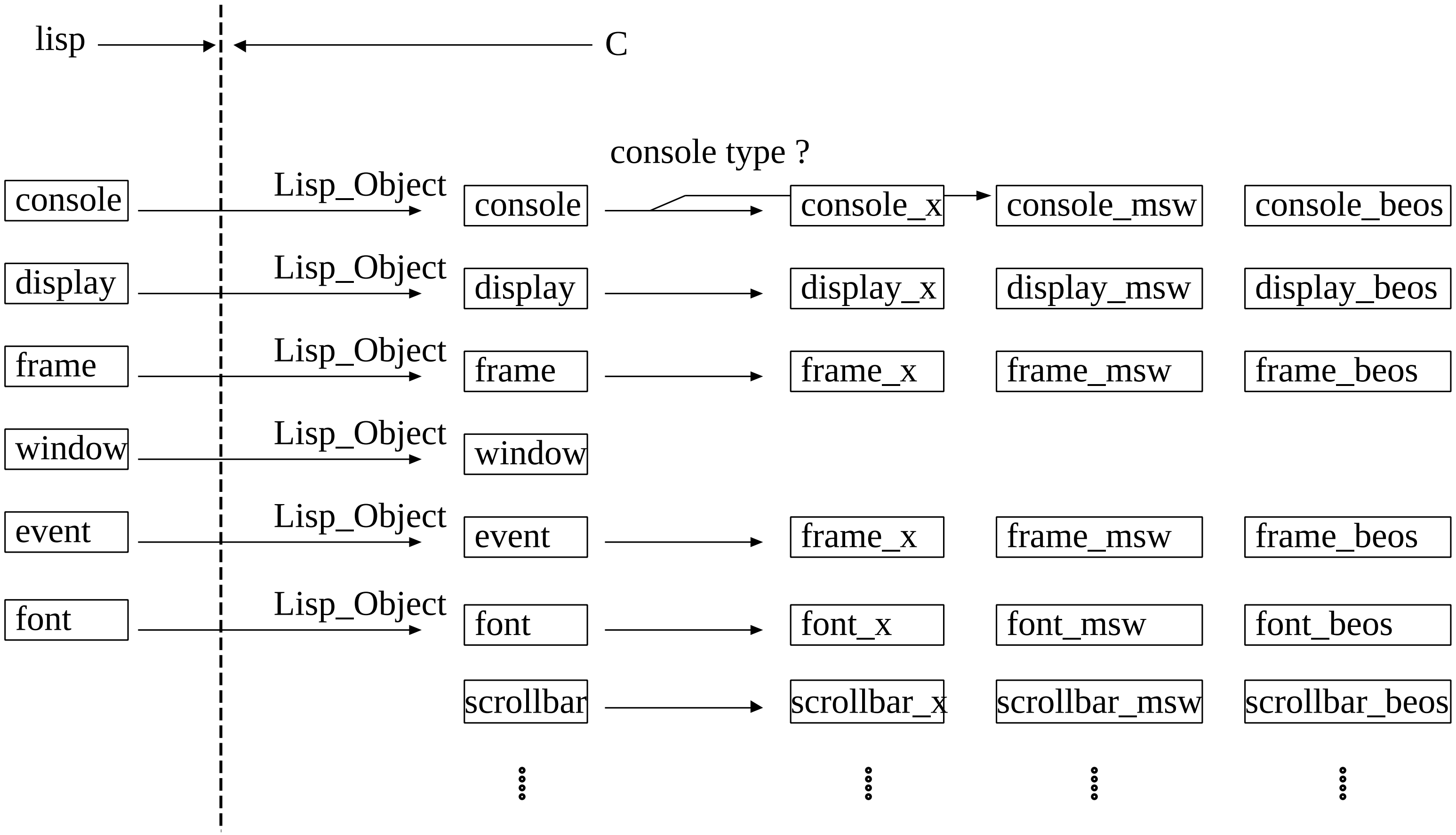
First native XEmacs window:
start simple



Case study: XEmacs

→ Is the app already multiplatform ?	YES
→ What kind of API does the app use ?	POSIX or WIN32
→ Type of the app ?	toolkit with abstraction layer
→ → Multi-platform already ?	YES
→ → Identify the complexity of toolkit usage	limited, main work is drawing text. Gui code is segmented.
→ → Disable complex controls ?	./configure --with-widgets=no \\ --with-scrollbars=no ...
→ App uses threads ?	NO
→ Is the work code rentrant ?	lisp engine reentrant to some extend, but very complex. We need to serialize input.

Object organization



lisp

C

console type ?

console → Lisp_Object → console → console_x → console_msw console_beos

display → Lisp_Object → display → display_x display_msw display_beos

frame → Lisp_Object → frame → frame_x frame_msw frame_beos

window → Lisp_Object → window

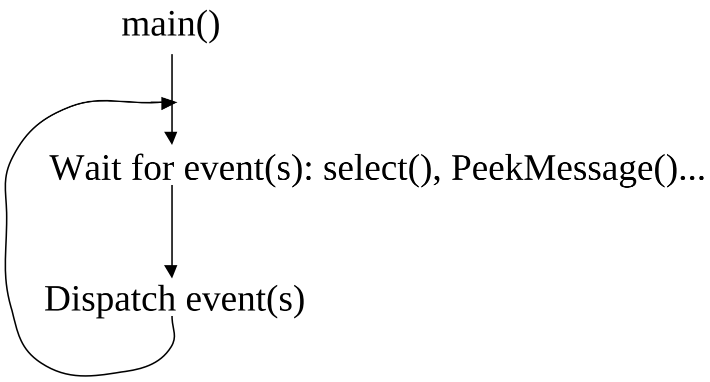
event → Lisp_Object → event → frame_x frame_msw frame_beos

font → Lisp_Object → font → font_x font_msw font_beos

scrollbar → scrollbar_x scrollbar_msw scrollbar_beos

Event handling

Monothreaded app:

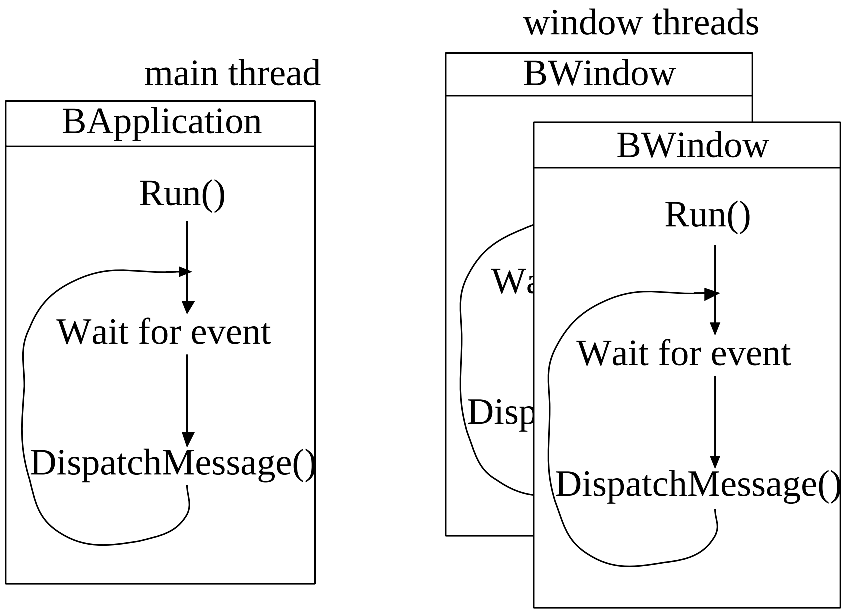


main()

Wait for event(s): select(), PeekMessage()...

Dispatch event(s)

BeOS app:



window threads

main thread

BApplication

Run()

Wait for event

DispatchMessage()

BWindow

Wa

Disp

BWindow

Run()

Wait for event

DispatchMessage()

Some questions:

How to run the app code if the main thread is executing BApplication::Run() ?

How can we get those messages from the main thread ?

Main thread

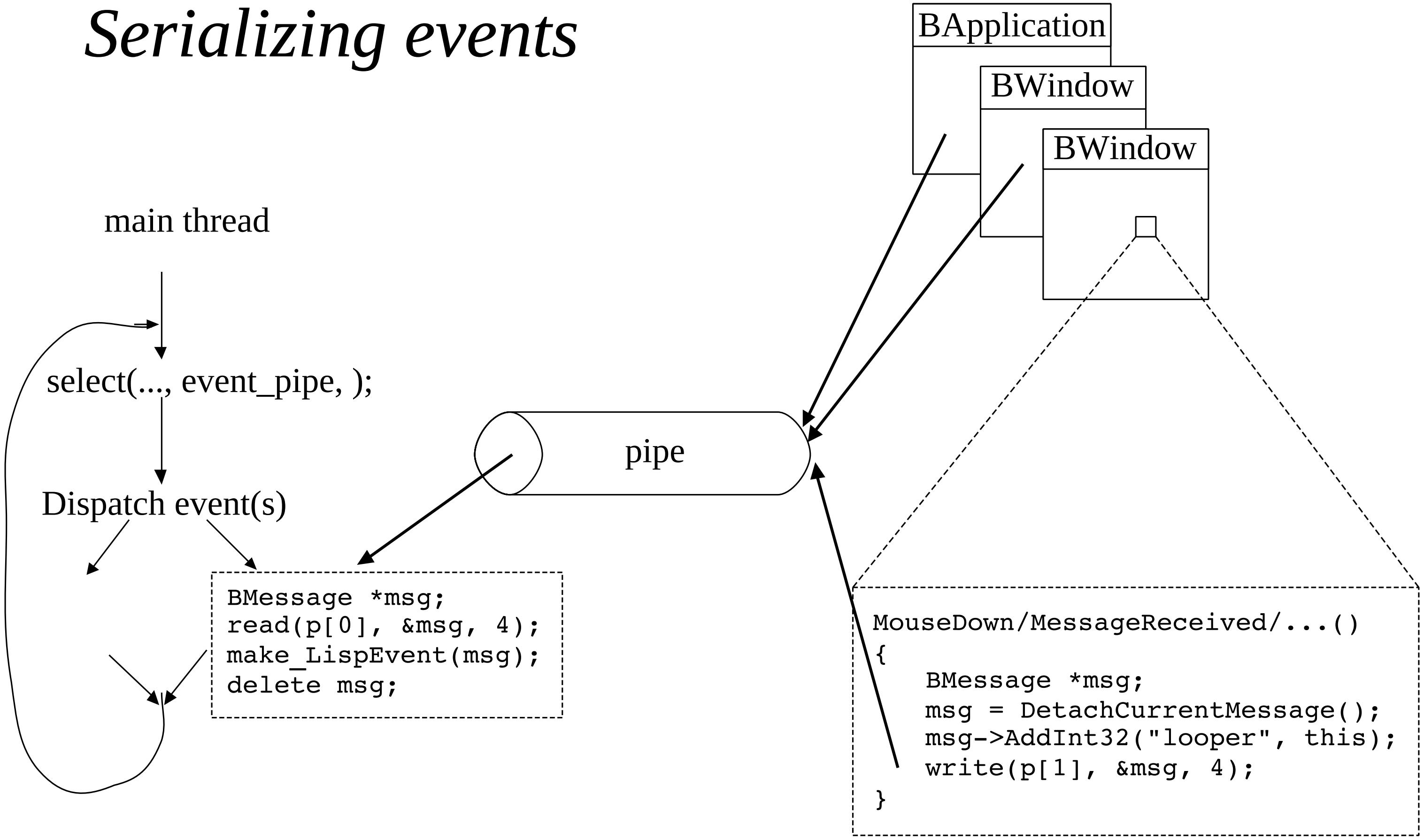
→ We need to run 2 codes concurrently...

→ Just use 2 threads !

BApplication takes over main thread...
actually the thread that calls Run().

```
run_bapp()
{
   be_app->Lock();
   be_app->Run();
   return 0;
}

main()
{
   new MyApplication("appli...");
   t = spawn_thread(run_bapp, "BApp", ...);
   resume_thread(t);
   be_app->Unlock();

   // app code
   ...

   be_app->Lock();
   be_app->Quit();
   delete be_app;

}
```

Serializing events



```
BMessage *msg;
read(p[0], &msg, 4);
make_LispEvent(msg);
delete msg;
```

```
MouseDown/MessageReceived/...()
{
   BMessage *msg;
   msg = DetachCurrentMessage();
   msg->AddInt32("looper", this);
   write(p[1], &msg, 4);
}
```

Some autoconf tricks

Cannot find libm ? `AC_CHECK_LIB(m,sin)` and remove hardcoded references

Detecting BONE libs

```
AC_CHECK_LIB(socket,socket)
AC_CHECK_LIB(nsl,gethostbyname)
AC_CHECK_LIB(resolv,gethostbyname)
AC_CHECK_LIB(bind,gethostbyname)
```

Undefined symbol 'syslog', 'openlog', ...

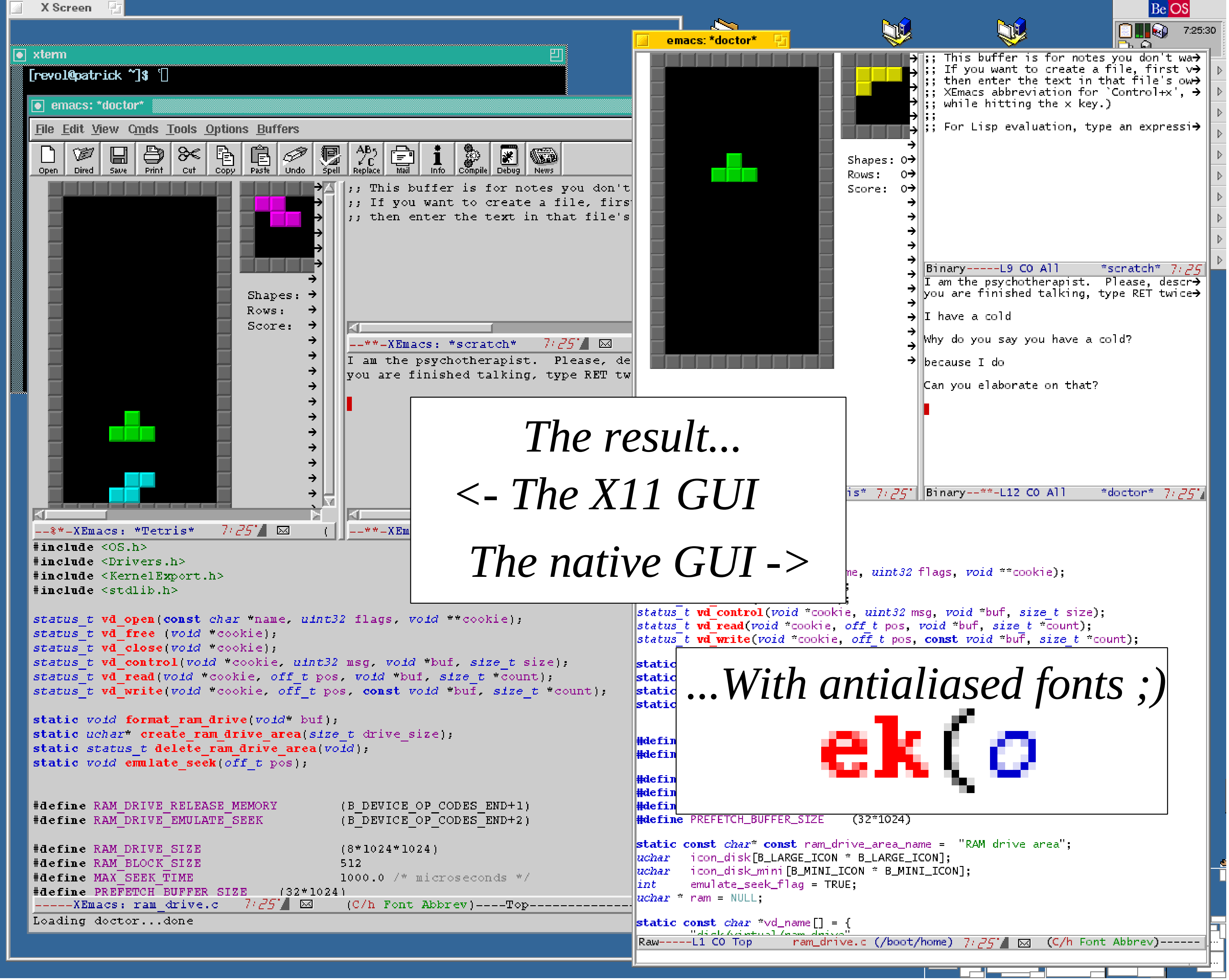
```
AC_CHECK_LIB(be,openlog)
```

function realpath, getpass, ... undefined:

```
AC_CHECK_FUNC(getpass)
```

then we need to ifdef out some code:

```
...
#ifdef HAVE_GETPASS
p = getpass("Password:");
#else
// some approximation, like:
char buff[50];
puts("Password:");
// pass appears in tty,
// but we get it
p = fgets(buff, 50, stdin);
#endif
...
```


X Screen
xterm
emacs: *doctor*
File Edit View Cmds Tools Options Buffers
Open Dired Save Print Cut Copy Paste Undo Spell Replace Mail Info Compile Debug News
Be OS
7:25:30
ek(o

The result...

<- The X11 GUI

The native GUI ->

...With antialiased fonts ;)

```
[revol@patrick ~]$
```

```
;; This buffer is for notes you don't
;; If you want to create a file, firs
;; then enter the text in that file's
--**-XEmacs: *scratch*     7:25
I am the psychotherapist.  Please, de
you are finished talking, type RET tw
```

```
Shapes:
Rows:
Score:
--%*-XEmacs: *Tetris*     7:25
```

```
#include <OS.h>
#include <Drivers.h>
#include <KernelExport.h>
#include <stdlib.h>

status_t vd_open(const char *name, uint32 flags, void **cookie);
status_t vd_free (void *cookie);
status_t vd_close(void *cookie);
status_t vd_control(void *cookie, uint32 msg, void *buf, size_t size);
status_t vd_read(void *cookie, off_t pos, void *buf, size_t *count);
status_t vd_write(void *cookie, off_t pos, const void *buf, size_t *count);

static void format_ram_drive(void* buf);
static uchar* create_ram_drive_area(size_t drive_size);
static status_t delete_ram_drive_area(void);
static void emulate_seek(off_t pos);


#define RAM_DRIVE_RELEASE_MEMORY        (B_DEVICE_OP_CODES_END+1)
#define RAM_DRIVE_EMULATE_SEEK          (B_DEVICE_OP_CODES_END+2)

#define RAM_DRIVE_SIZE                  (8*1024*1024)
#define RAM_BLOCK_SIZE                  512
#define MAX_SEEK_TIME                   1000.0 /* microseconds */
#define PREFETCH_BUFFER_SIZE        (32*1024)
-----XEmacs: ram_drive.c    7:25      (C/h Font Abbrev)----Top-------------
Loading doctor...done
```

```
;; This buffer is for notes you don't wa
;; If you want to create a file, first v
;; then enter the text in that file's ow
;; XEmacs abbreviation for `Control+x',
;; while hitting the x key.)
;;
;; For Lisp evaluation, type an expressi
Binary-----L9 C0 All       *scratch*  7:25
I am the psychotherapist.  Please, descr
you are finished talking, type RET twice

I have a cold

Why do you say you have a cold?

because I do

Can you elaborate on that?
Binary--**-L12 C0 All      *doctor*  7:25
```

```
Shapes: 0
Rows:   0
Score:  0
is*  7:25
```

```
me, uint32 flags, void **cookie);

status_t vd_control(void *cookie, uint32 msg, void *buf, size_t size);
status_t vd_read(void *cookie, off_t pos, void *buf, size_t *count);
status_t vd_write(void *cookie, off_t pos, const void *buf, size_t *count);

static
static
static
static

#defin
#defin

#defin
#defin
#defin
#define PREFETCH_BUFFER_SIZE    (32*1024)

static const char* const ram_drive_area_name =  "RAM drive area";
uchar   icon_disk[B_LARGE_ICON * B_LARGE_ICON];
uchar   icon_disk_mini[B_MINI_ICON * B_MINI_ICON];
int     emulate_seek_flag = TRUE;
uchar * ram = NULL;

static const char *vd_name[] = {
Raw-----L1 C0 Top     ram_drive.c (/boot/home)  7:25    (C/h Font Abbrev)-------
```

Questions ?

Thanks for coming.
Any question ?