Current status of 64-bit IRAF

Chisato Yamauchi

Institute of Space and Astronautical Science, Japan Aerospace Exploration Agency
(ISAS/JAXA)

Overview

- Importance of 64-bit IRAF
- Difficulty of porting IRAF to 64-bit
- SPP data models
- Our project - The IRAF64 Project
- Our results
- Screenshots
- Our Plan

Importance of 64-bit IRAF

- IRAF is the de fact standard software for analysis of optical/infrared data.
- Modern large telescopes and satellites still utilize the IRAF.
- IRAF will keep being used in the future.
- Current 32-bit IRAF has the 4GB barrier, which should be removed to handle incredibly high-resolved/wide-field data in the future.

  (e.g. SUBARU Hyper Suprime-Cam will output a 2.8GB of FITS)

next: Difficulty of porting IRAF to 64-bit

Difficulty of porting IRAF to 64-bit

- IRAF requires a small assembler code for each architecture.

```
zsvjmp_:
    # %rsi ... &status   %rdi ... &jumpbuf
    movq   %rsi, (%rdi)   # store &status in jmpbuf[0]
    movl   $0, (%rsi)     # zero the value of status
    addq   $8, %rdi       # change point to &jmpbuf[1]
    movl   $0, %esi       # change arg2 to zero
    jmp    __sigsetjmp    # let sigsetjmp do the rest
```

This is zsvjump.s for x86_64 OS. The arguments of a function are received by registers (not by stack).

Difficulty of porting IRAF to 64-bit

- Intrinsic problem is the tacit assumption of size of integer and pointer types in IRAF code.

  i.e., `sizeof(int) == sizeof(long) == sizeof(pointer) == sizeof(real).`

  Examples of actual problems:

- Incompatible pointer args of functions.
- There is ‘Memi’ (for memory access for integer type). However, ‘Memp’ (that for pointer) does not exist.
- P2R() and P2I() macros do not exist .
- There are tricks around ‘struct’ constructions in some codes.

next: Data model of SPP

Data model of SPP

- Which data model is suitable for 64-bit SPP?

	short	int	long	pointer	
ILP32	16	32	32	32	(32-bit gcc)
LP64	16	32	64	64	(64-bit gcc)
ILP64	16	64	64	64	

  - If we select ILP64, a number of tools will work without code modification. However, IRAF contains many codes dependent on 32-bit integer.
  - Genuine approach is to renounce the tacit assumption of integer/pointer sizes.

We select LP64 model and revise SPP specifications, and remove the root problem.

next: Our project

Our project - The IRAF64 Project

- Porting the IRAF software to the x86_64 Linux OS with NATIVE x86_64 (Opteron,EM64T) executable.
- This IRAF is developed as a candidate of IRAF Revision 3.0.
- Our project uses the SourceForge.Net (https://sourceforge.net/).
- The source code is maintained using the Subversion, and the repository is public now.
- We review and update all C/SPP codes.

Our project - The IRAF64 Project

- Project members

  - Project Admins/Developer:

    Chisato Yamauchi (Japan Aerospace Exploration Agency)

  - Developer/Tester:

    Keith Rosema (University of Washington)
    Sergio Pascual (Complutense University of Madrid)
    Yasushi Nakajima (National Astronomical Observatory of Japan)

next: Our results

Our results: general improvements

- Improvements of C code to meet the present standard.
  - Enabled gcc's '-Wall' option and minimized gcc's warnings.
  - Replaced ancient style (K&R -> ANSI).
  - Appended complete prototype declarations.
  - Improved a lot of security issues.
  - gcc-4.3 is also OK!
- Rearranged IRAF directory structure.
- Redefined 'MACH' environment variable.

  e.g., MACH=x86_64-linux-redhat, MACH=i386-linux-generic, etc.

Our results: general improvements

- Appended an mechanism of auto-detection of OS and architecture when building IRAF.
- Applied ‘GNU make’ to build IRAF easily.

```
$ make boot
$ make iraf
$ make reboot
$ make tables
$ make noao
$ make install (DESTDIR=...)
```

  OS and architecture are automatically detected on ‘`make boot`’ stage.

- Replaced csh-scripts with sh-scripts.

Our results: updates for x86_64

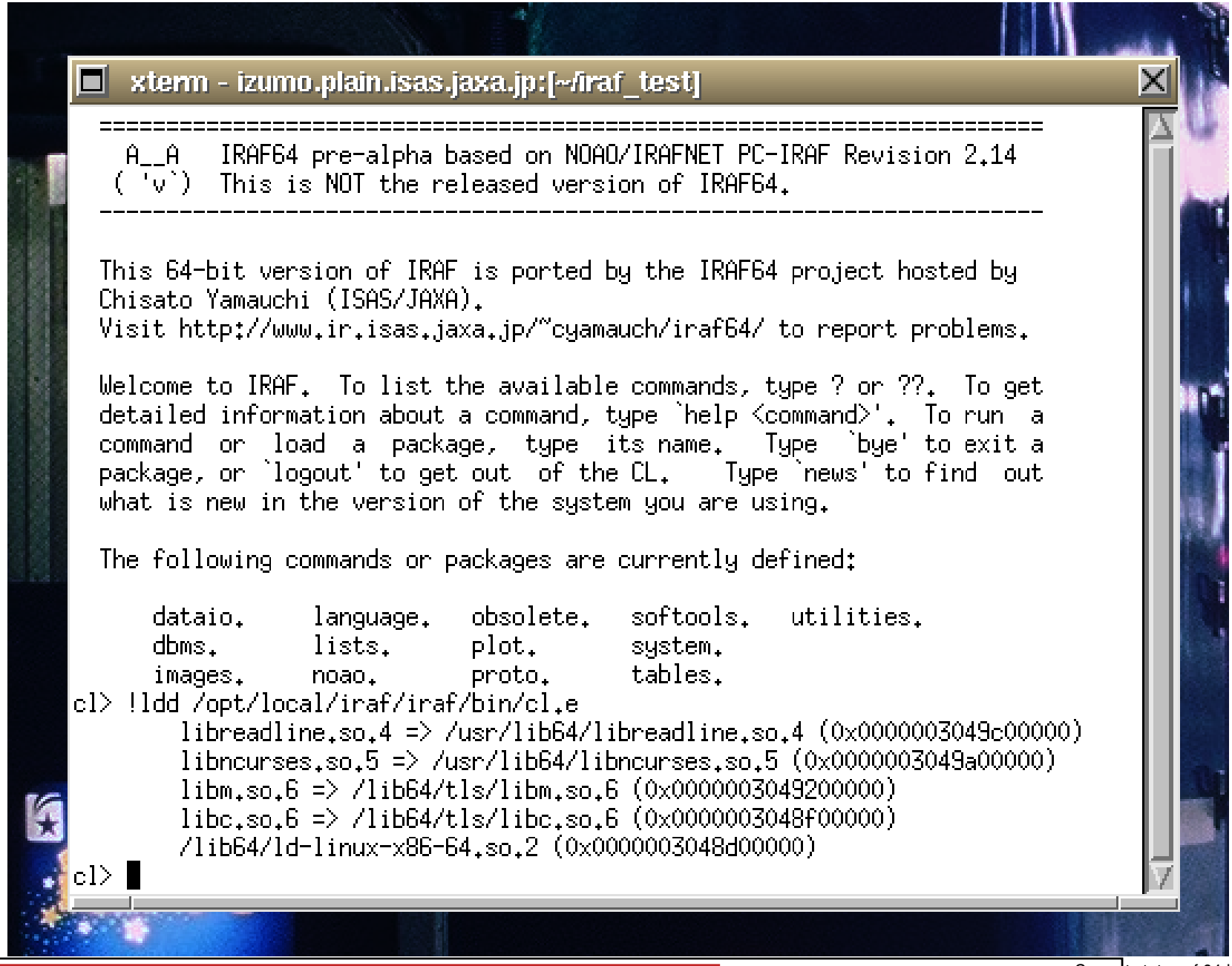
- Wrote an assembler code for Linux x86_64 architecture.
- Prepared the environment for 'LP64-SPP' development.
  - Some modifications for SPP preprocessor/convertor and f77 configurations.
  - xc, rpp, xpp and mkpkg are OK!
- Reviewed and updated C and SPP codes for libsys.a, libvops.a, libcur.a, libstg.a, libds.a, libex.a and libc.a.
  - Supported 64-bit memory allocation and file I/O.
  - Supported large data ($>$ 4GB) in the majority of functions.

Our results: updates for x86_64

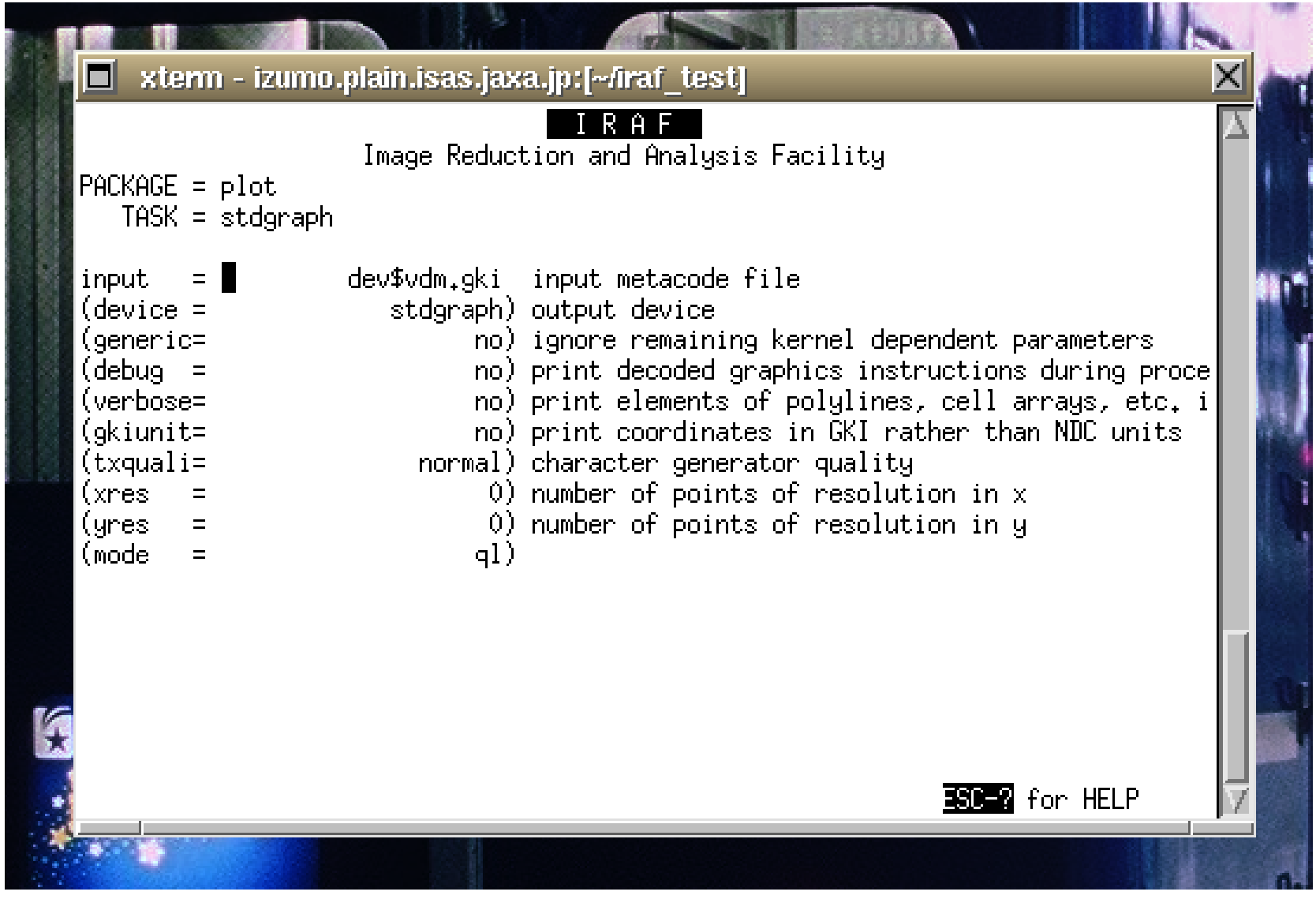
- ‘cl’ is running on CentOS 4.x/Fedora 8,9.
  - A few tasks of core system are OK.
  - However, the majority of tasks cause PANIC error.

next: Screenshots

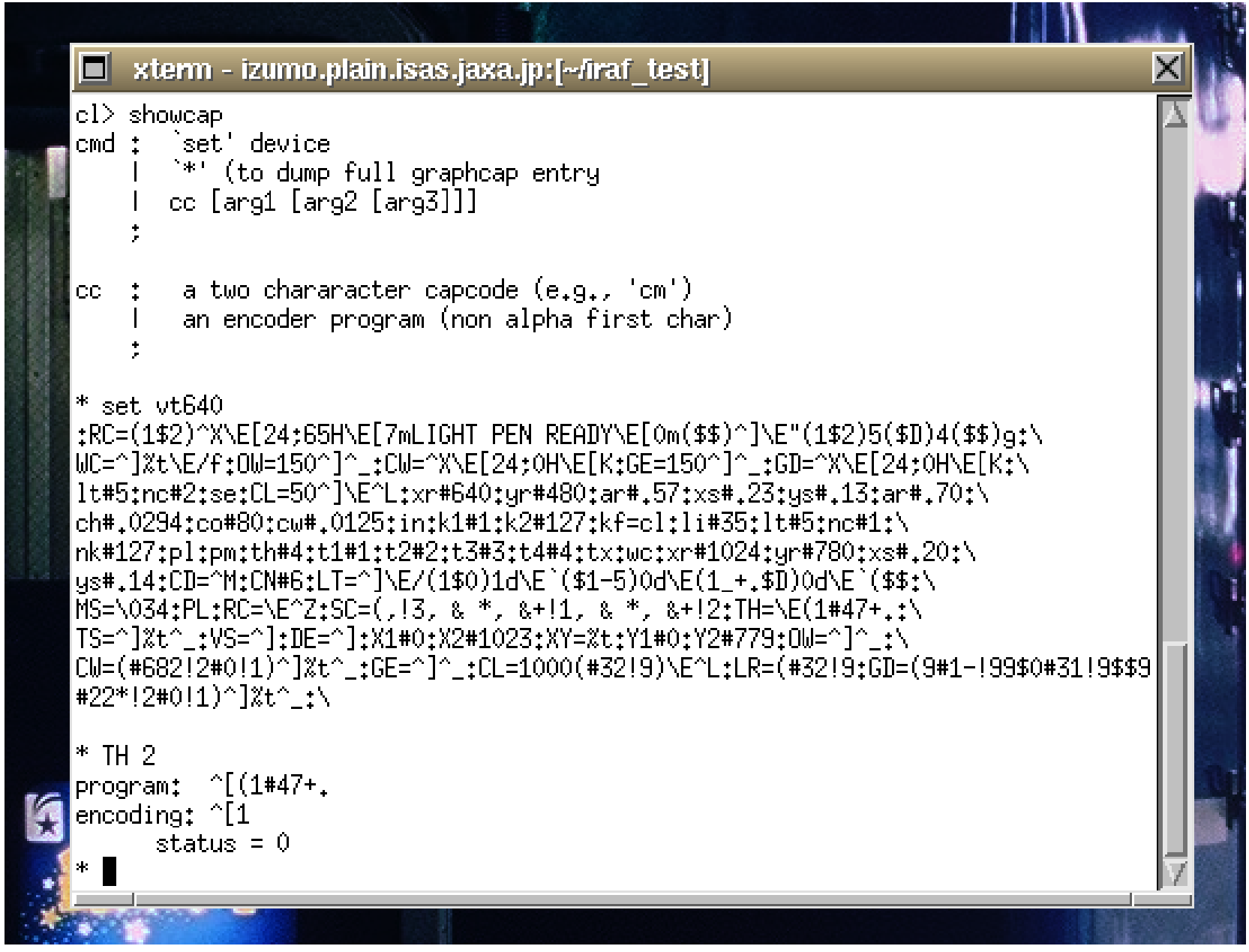
Screenshots (cl)



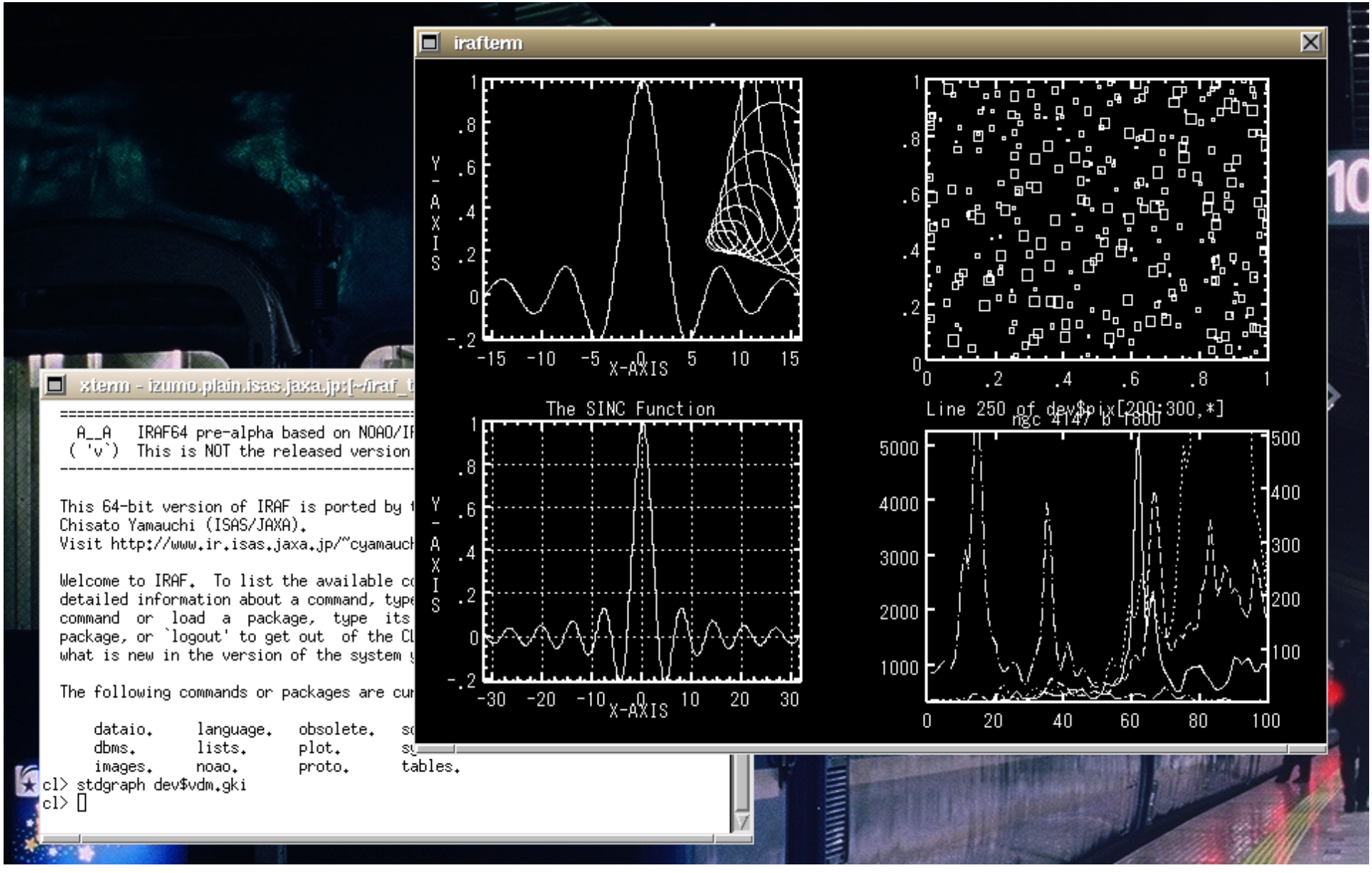
Screenshots (epar)



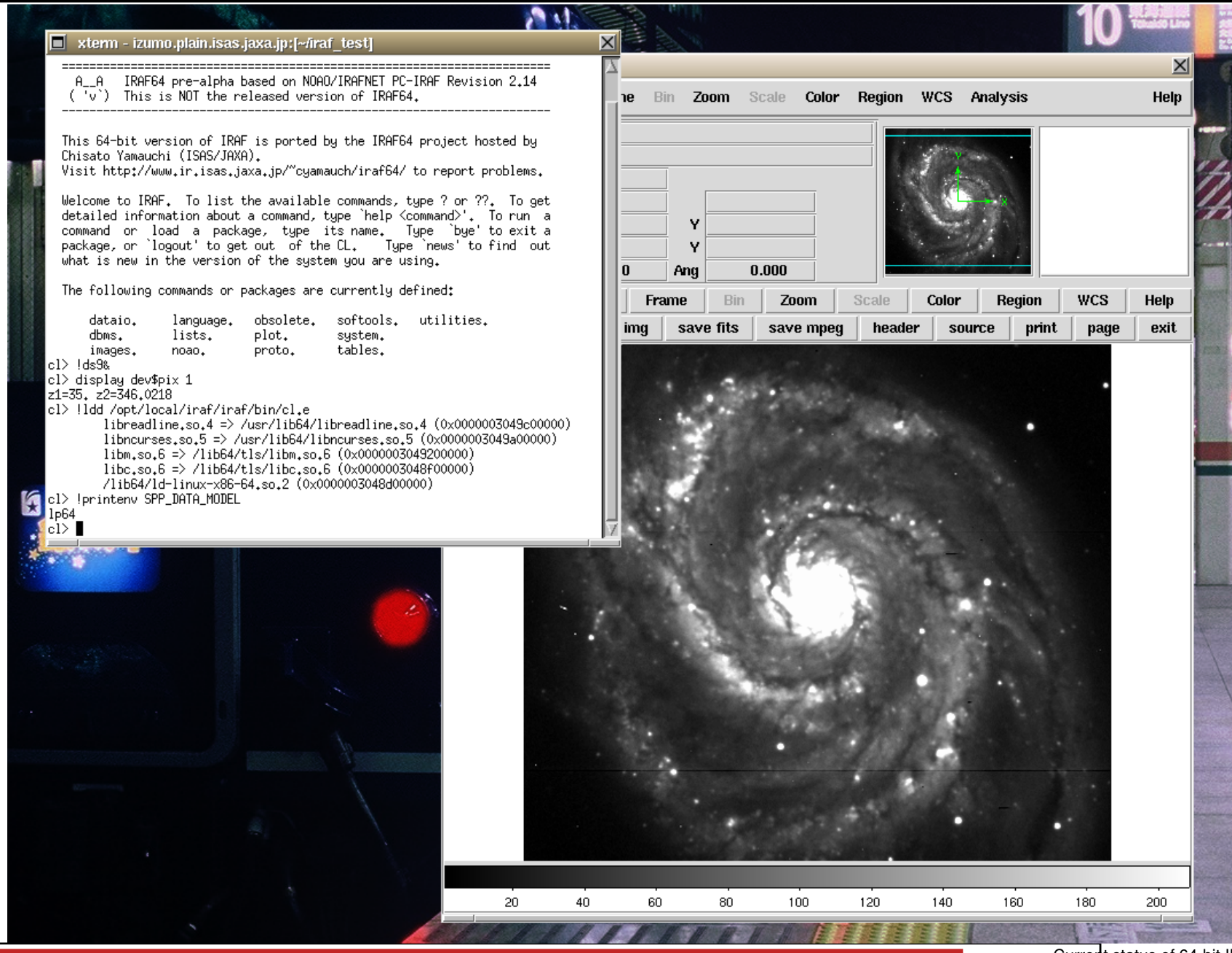
Screenshots (showcap)



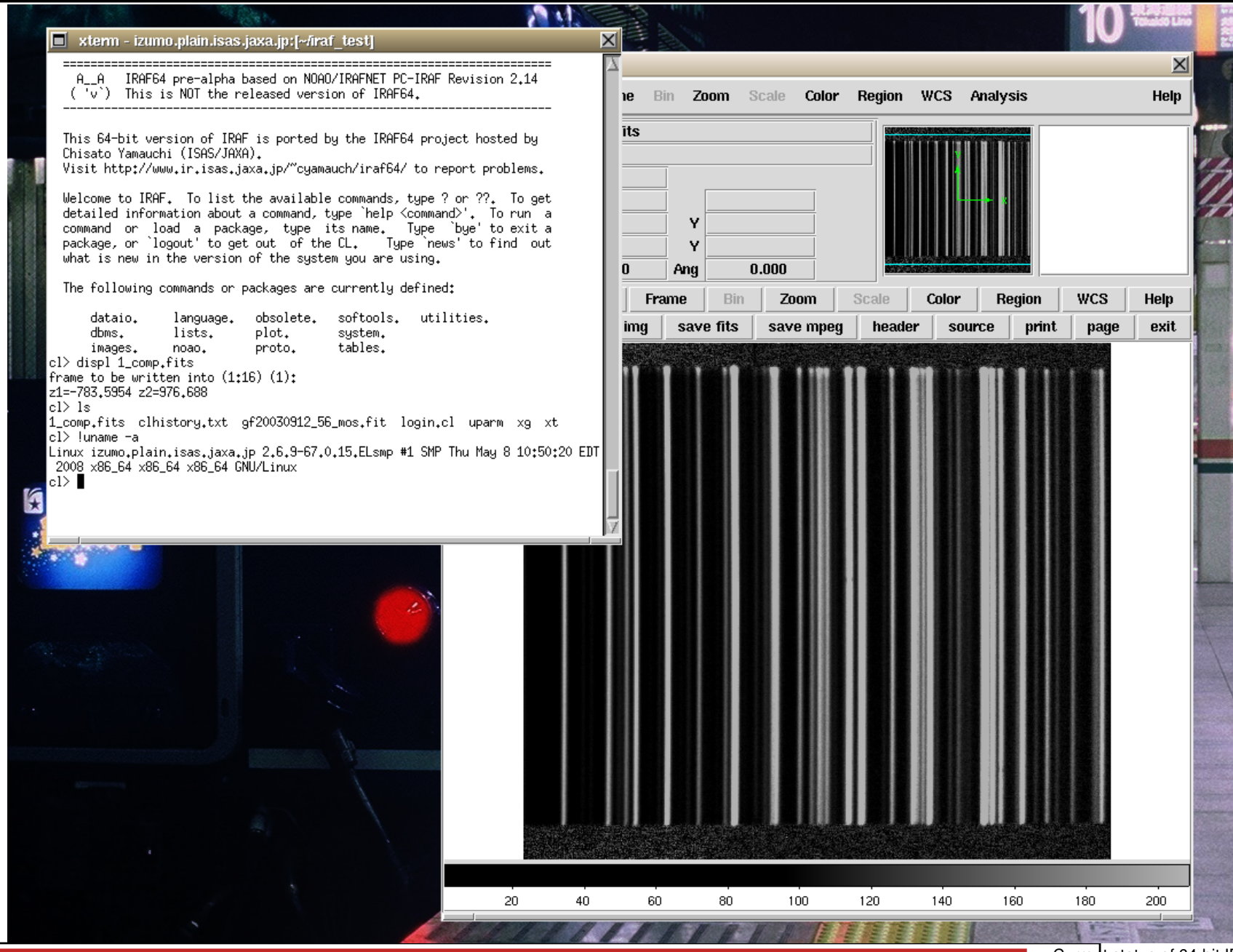
Screenshots (stdgraph)



Screenshots (display)



Screenshots (FITS file)



Our plan

- First, we intend to provide 64-bit support of core system.
  (i.e., libraries and basic tasks in sys/ pkg/ and math/ directories).
- Tables library and NOAO standard utilities will be updated after support of core system.
- Collaboration with STScI/NOAO may be wanted for updates of Tables library and NOAO standard utilities.