Tim Liu

May 20, 1997

NIRSPEC Software Programming Note 09.01
Program Building

1 Introduction

This document describes how to build the **NIRSPEC** server and client programs. In this draft version we discuss the compilation of C programs which include the RPC server, keyword library, DataViews-based **GUI**, **CLI**, echelle format simulator (**EFS)** gateway program, image rotator **GUI**, and shareable libraries for IDL interface routines. In future versions we will also include the building procedures for the IDL-based programs such as the quick-look (**QL**), **EFS GUI**, data reduction pipeline, and S/N estimator. The compilation of occam code for the transputer system will also be discussed in the future.

2 Building Server Program

The RPC server code is located in **/kroot/kss/nirspec**. In addition, the **NIRSPEC** keyword library and RPC-KTL interface routines are also in this directory. The **NIRSPEC** design requires a simulation mode which runs without the instrument hardware. This can be achieved by connecting the client programs to a simulation server which doesn't talk to the underlying transputer system and therefore no real action is taken by the transputer-controlled instrument. The real instrument server and simulation server are built by two separate make files as discussed in the following.

2.1 Instrument server

To build the instrument (non-simulation mode) server program, go to the server program directory **/kroot/kss/nirspec** and type:

```
% make
```

This will execute the make file **makefile** and produce the keyword shareable library **libnirspec_keyword.so.0.0** and the server executable **nirspec_server**. In addition, **makefile** invokes the RPC protocol compiler **rpcgen** to compile the protocol file **nrpc.x**. **rpcgen** automatically generates the header file **nrpc.h**, the server and client stub programs **nrpc_svc.c** and **nrpc_clnt.c**, and the external data representation code

nrpc_xdr.c. The module **nrpc_svc.c** is not used by the **NIRSPEC** software, but only serves as a template for writing the server main program **nprc_server.c** which requires an asynchronous I/O loop not provided by any **rpcgen**-generated RPC server stub code. The server make file **makefile** is listed below:

```
##############################################################################
#+
#  Author:    Tim Liu
#
#  Date:      10/05/95
#
#  Description: makefile to build
#             libnirspec_keyword.so.0.0  - keyword sharable library
#             nirspec_server             - RPC server program
#-
##############################################################################
APP    = nrpc
CC     = cc
CFLAGS = -g -I$(KROOT)/rel/default
LIBS   = -lnsl -lc -lm -lucb \
         /kroot/rel/default/lib/libktl.so.0.0 \
         /kroot/rel/default/lib/libktlker.so.0.0
RM     = rm -f

TARGET = libnirspec_keyword.so.0.0 nirspec_server clean
all: $(TARGET)

# Build keyword sharable library
libnirspec_keyword.so.0.0: nirspec_keyword.o $(APP)_face.o $(APP)_clnt.o \
                           $(APP)_xdr.o utils.o
       ld -G -o $@ nirspec_keyword.o $(APP)_face.o $(APP)_clnt.o \
       $(APP)_xdr.o utils.o $(LIBS)

# Build RPC server program
nirspec_server: $(APP)_server.o $(APP)_svc_proc.o $(APP)_xdr.o fileio.o tspcom.o
       $(CC) -o nirspec_server $(APP)_server.o $(APP)_svc_proc.o \
       $(APP)_xdr.o fileio.o tspcom.o utils.o $(LIBS) /opt/transtech/lib/libtsp.a

# Compile the protocol
$(APP)_clnt.c $(APP)_svc.c $(APP)_xdr.c $(APP).h: .rpcgen
.rpcgen: $(APP).x
       rpcgen $(APP).x
       touch .rpcgen

clean:
       $(RM) $(APP)_face.o $(APP)_server.o

nirspec_keyword.o:  nirspec.h error.h
$(APP)_face.o:      $(APP).h nirspec.h error.h
$(APP)_server.o:    $(APP).h nirspec.h error.h
$(APP)_svc_proc.o:  $(APP).h nirspec.h error.h
fileio.o:           $(APP).h nirspec.h error.h
tspcom.o:           tspcom.h
utils.o:            nirspec.h
```

Note that the object files **nrpc_face.o** and **nrpc_server.o** must be deleted after the compilation because they are server-mode (simulation or non-simulation) dependent. If the simulation server make file **makefile_sim** links with the two objects generated by the instrument server make file **makefile**, an incorrect server executable will be produced.

2.2 Simulation server

The simulation server make file **makefile_sim** defines the macro preprocessing directive **SIMULATION** as shown below:

```
CC = cc -DSIMULATION
```

This macro is used by **nrpc.h** and **nrpc_server.c**. In addition, **nrpc_face.c** is also affected by **SIMULATION** through **nrpc.h**. This is why both **nrpc_face.c** and **nrpc_server.c** must be re-compiled each time **makefile** or **makefile_sim** is invoked. Except for a few naming differences, **makefile_sim** is similar to **makefile**. To build the simulation server, just type:

```
% make -f makefile_sim
```

and the make file will generate the keyword library **libnirspecsim_keyword.so.0.0** and the simulation server executable **nirspecsim_server**.

2.3 Symbolic links

The **NIRSPEC** server run-time environment requires symbolic links to the server startup C-shell script **run_nirspec_server** and the keyword shareable object libraries **libnirspec_keyword.so.0.0** and **libnirspecsim_keyword.so.0.0** in the directory **/kroot/rel/default/bin**. You make these links only once as long as you don't delete them later. If the link to **nirspec_server** already exists, you'll see the following when using the UNIX command "**which**" to check its existence:

```
nirspec% which run_nirspec_server
/kroot/rel/default/bin/run_nirspec_server
```

Check the other links the same way. If the link is not there, you can set it up such that:

```
nirspec% cd /kroot/rel/default/bin
nirspec% ln -s /kroot/kss/nirspec/run_nirspec_server
```

Try the same for the other links if they don't exist.

2.4 Server initialization file

The server software reads **NIRSPEC** environmental variables from an initialization file named **NirspecServerInit** in **/kroot/kss/nirspec** which is listed below:

```
#
# NIRSPEC server program initialization
#

# Set environment variables
setenv NIRSPEC_INSTR_LOG       /tmp/NirspecInstr.log
setenv NIRSPEC_CONFIG_FILE     /tmp/Nirspec.cfg
setenv NIRSPEC_BOOT_FILE       $KROOT/kss/nirspec/trans/nirspec.btl
setenv NIRSPEC_HEADERINFO_SPEC $KROOT/kss/nirspec/header_info.nirspec_spec
setenv NIRSPEC_HEADERINFO_SCAM $KROOT/kss/nirspec/header_info.nirspec_scam
setenv NIRSPEC_SERVER_DIR      $KROOT/kss/nirspec/
setenv NIRSPEC_TMPFS_DIR       /tmp/
```

This file must exist in order for the server programs to run.

2.5 Client access control file

The server computer maintains a user access control file called **Authorize** in **/kroot/kss/nirspec/** which includes all the clients that have permission to access the server. The server program will look up this file to decide whether to accept a client connection request. Here's a sample of this authorization file:

```
# NIRSPEC authorization file
# user:host:access
nirspec:crab.astro.ucla.edu:1
nirspec:jupiter.astro.ucla.edu:2
nirspec:kapoho.keck.hawaii.edu:2
```

The first two fields assign a valid login and host machine name and the third one gives the user access right. The access right "**1**" is assigned to the principle user with unlimited access rights to the instrument server and "**2**" is assigned to a secondary user with limited rights. Edit this file to add or remove a client. Make sure **Authorize** has the permission mode "**0644**", i.e., the owner has both read and write permission while the group and other have read permission only. This file should be owned by the server login "**nirspec**" or the super-user.

2.6 Syslog setup

The **NIRSPEC** software uses the UNIX **syslog** function to log instrument system messages both into the log file **/var/log/nirspec.log** and on the console **cmdtool**. You

need to do a setup the first time you build the **NIRSPEC** so that the daemon **syslogd** runs in a new configuration. Below is the **syslog** configuration procedure:

a) Log into the server machine as root and add the following lines to the end of the **syslog** configuration file **/etc/syslog.conf**:

```
local0.err;local0.warning;local0.info     /dev/console
local0.err;local0.warning;local0.info     /var/log/nirspec.log
```

b) Create a log file named **nirspec.log** in **/var/log** and allow **rw** permission such that:

```
nirspec% touch nirspec.log
nirspec% chmod 666 nirspec.log
```

c) Send a "**-HUP**" signal to **syslogd**:

```
nirspec% ps -ef | grep syslogd
nirspec% kill -HUP pid
```

d) Confirm that **syslog** is updating the log file **nirspec.log** by checking its size or contents when the **NIRSPEC** program is running. Also check message logging from the **cmdtool** console.

Once in a while, you should remove the old contents in **nirspec.log** when it becomes too big. But, remember *not* to delete this file because you'll have to create another one using the procedure listed above if you deleted it.

2.7 Server startup script

The server program is launched by the C-shell script **run_nirspec_server** in **/kroot/kui/xnirspec** as listed below:

```
#!/bin/csh -f
#
# Script to launch NIRSPEC server program
#
# Tim Liu 5/9/97

# set default values
set f1 = ''
set f2 = ''
set simulate = 0

# check arguments
```

```
foreach arg($*)
    switch ($arg)
        case "-s":
            set simulate = 1
            breaksw
        case "-r":
            set f1 = $arg
            breaksw
        case "-nodcs":
            set f2 = $arg
            breaksw
        default:
            echo 'usage: run_nirspec_server [-s] [-r] [-nodcs]'
            goto QUIT
    endsw
end

# run server program
if ( $simulate ) then
    ps -ef | grep nirspecsim_server > /dev/null
    if ( $status == 0 ) then
        echo 'simulation server is already running'
    else
        exec /kroot/kss/nirspec/nirspecsim_server
    endif
else
    ps -ef | grep nirspec_server > /dev/null
    if ( $status == 0 ) then
        echo 'server is already running'
    else
        exec /kroot/kss/nirspec/nirspec_server $f1 $f2
    endif
endif
```

As can be seen, the script will check whether the server is already running first.

3 Building Client Programs

3.1 Program compilation

The **NIRSPEC** C client programs are located in two places. The **GUI**, **CLI**, **EFS** gateway, and IDL interface routine libraries are in **/kroot/kui/xnirspec**, while the image rotator software is located in /**kroot/kui/ximrot**. The client building procedure is handled by **makefile** in either directory. To compile the **NIRSPEC** client programs in the directory **/kroot/kui/xnirspec**, type:

% make

And this will produce three client executables: **xnirspec**, **cnirspec** and **efs_gateway**, and two shareable object libraries: **ql_client.so** and **efs_client.so**. The make file **makefile** is listed below:

```
##############################################################################
#+
#  Author:    Tim Liu
#
#  Date:      10/25/95
#
#  Description: makefile to build
#             xnirspec      - GUI program
#             cnirspec      - CLI program
#             efs_gateway   - gateway to EFS
#             ql_client.so  - quick-look client shareable object library
#             efs_client.so - EFS client shareable object library
#-
##############################################################################
INCLUDE      = /usr/local/dv/include

CC           = cc
CFLAGS       = -g -I$(INCLUDE) -I$(KROOT)/rel/default/include
LDFLAGS      = -o xnirspec

NAMES        = xnirspec dataviews create_interest callbacks misc \
               cli_server ql_server dcs_util socket
SOURCE       = $(NAMES:%=%.c)
OBJECT       = $(NAMES:%=%.o)

NAMES2       = efs_gateway efs_server socket
SOURCE2      = $(NAMES2:%=%.c)
OBJECT2      = $(NAMES2:%=%.o)

LIBS         = -L$(KROOT)/rel/default/lib \
               -lktl -lktlker -lkcl -ldl -lXm -lucb
LIBS2        = -ltcl7.5 -lsocket -lm -ldl -lnsl -lreadline -ltermcap
LIBS3        = -L$(KROOT)/rel/default/lib \
               -lktl -lktlker -lkcl -ldl -lsocket -lucb -lm

TARGET       = xnirspec cnirspec ql_client.so efs_gateway efs_client.so
all: $(TARGET)

# Build GUI application
xnirspec: $(OBJECT)
       DVlink $(LDFLAGS) $(OBJECT) $(LIBS)

# Build CLI application
cnirspec: cli.o socket.o
       $(CC) -o cnirspec cli.o socket.o $(LIBS2)

# Build QL socket client sharable object library
ql_client.so: ql_client.o
       ld -G -o $@ ql_client.o socket.o

# Build EFS gateway program
efs_gateway: $(OBJECT2)
```

```
	$(CC) -o efs_gateway $(OBJECT2) $(LIBS3)

# Build EFS socket client sharable object library
efs_client.so: efs_client.o
	ld -G -o $@ efs_client.o socket.o

xnirspec.o:         xnirspec.h
dataviews.o:        xnirspec.h dataviews.h
create_interest.o:  xnirspec.h
callbacks.o:        xnirspec.h
misc.o:             xnirspec.h nirspec.h
cli.o:              nirspec.h
cli_server.o:       xnirspec.h nirspec.h
ql_server.o:
ql_client.o:
efs_gateway.o:      nirspec.h xnirspec.h
efs_server.o:       nirspec.h
efs_client.o:
dcs_util.o:         xnirspec.h
socket.o:
```

Note that **`DVlink`** is a shell script for compiling DataViews-based programs. **`DVlink`** generates debug information. To create the final version of a program without symbolic debug information, use **`DVlink2`** which employs optimized DV libraries in the compilation.

Similarly, to build the image rotator GUI **`ximrot`**, go to **`/kroot/kui/ximrot`** and type "**`make`**". The corresponding make file **`makefile`** is listed below:

```
##############################################################################
#+
#  Author:    Tim Liu
#
#  Date:      07/10/96
#
#  Description: makefile to build
#             ximrot   - image rotator program
#-
##############################################################################
INCLUDE      = /usr/local/dv/include

CFLAGS       = -g -I$(INCLUDE) -I/kroot/rel/default/include
LDFLAGS      = -o ximrot

NAMES        = ximrot dv create_interest callbacks
SOURCE       = $(NAMES:%=%.c)
OBJECT       = $(NAMES:%=%.o)

LIBS         = -lXm -lXt -lX11 -lm -L$(LIBDIR) -ldl \
               /usr/ucblib/libucb.so.1 \
               /kroot/rel/default/lib/libktl.so.0.0 \
               /kroot/rel/default/lib/libktlker.so.0.0 \
               /kroot/kui/xnirspec/socket.o

TARGET       = ximrot
all: $(TARGET)
```

```
ximrot: $(OBJECT)
       DVlink $(LDFLAGS) $(OBJECT) $(LIBS)

ximrot.o:     ximrot.h
dv.o:         ximrot.h dv.h
```

3.2 Symbolic link

Check whether the symbolic link to the **NIRSPEC** client startup C-shell script file **/kroot/kui/xnirspec/run_nirspec_client** has been created in the **/kroot** binary release directory **/kroot/rel/default/bin**. If not, try the following:

```
nirspec% cd /kroot/rel/default/bin
nirspec% ln -s /kroot/kui/xnirspec/run_nirspec_client
```

3.3 Client initialization file

The initialization file **NirspecClientInit** in **/kroot/kui/xnirspec** defines several environmental variables as shown below:

```
#
# NIRSPEC client program initialization
#

# Set environment variables
setenv NIRSPEC_SERVER_HOST    crab.astro.ucla.edu
setenv NIRSPEC_SOCKET_DIR     /tmp/
setenv NIRSPEC_SCRIPT_DIR     /kroot/kui/xnirspec/scripts/
setenv NIRSPEC_CONFIG_DIR     /kroot/kui/xnirspec/
#setenv NIRSPEC_DV_DIR        $KROOT/kui/xnirspec/
#setenv IMROT_DV_DIR          $KROOT/kui/ximrot/
```

This file must exist in order for the client programs to run. As can be seen, the client initialization file needs to be modified when reconfiguring a new client machine.

3.4 Client startup script

The **NIRSPEC** client software consists of several stand-alone programs and they are launched by the C-shell script **run_nirspec_client** in **/kroot/kui/xnirspec** as listed below:

```
#!/bin/csh -f
#
# Script to launch NIRSPEC client programs
#
# Tim Liu 1/14/96
```

```
# set default values
set f1 = ''
set f2 = ''
set f3 = ''
set noql = 0
set noimrot = 0
set noefs = 0

# check arguments
foreach arg($*)
    switch ($arg)
        case "-s":
            set f1 = $arg
            breaksw
        case "-nodcs":
            set f2 = $arg
            breaksw
        case "-noql":
            set f3 = $arg
            set noql = 1
            breaksw
        case "-noimrot":
            set noimrot = 1
            breaksw
        case "-noefs":
            set noefs = 1
            breaksw
        default:
            echo 'usage: run_nirspec_client [-s] [-nodcs] [-noql] [-noimrot]
                  [-noefs]'
            goto QUIT
    endsw
end

# run xnirspec
cd /kroot/kui/xnirspec
exec ./xnirspec $f1 $f2 $f3 &

# run EFS gateway program
if ( !($noefs) ) then
    exec ./efs_gateway $f1 &
endif

# run ximrot
if ( !($noimrot) ) then
    cd /kroot/kui/ximrot
    exec ./ximrot $f1 $f3 &
endif

# run idl quick-look
if ( !($noql) ) then
```

```
    source /kroot/kui/qlook/idl_setup
    xrdb -merge /crab/home1/nirspec/.Xdefaults
    xterm -iconic -e idl -server &
    sleep 10
    /kroot/kui/qlook/rpc/quicklook
    /kroot/kui/qlook/rpc/exitidl
endif

QUIT:
```

Note that the execution flags in **`run_nirspec_client`** correspond to program arguments in the client software. When the arguments of a client program is changed, the correspondent execution options in the script file should also be modified.