darc.ing.puc.cl

Operating instructions

Note - for some reason the dns server doesn't seem to work... if this persists, then when you boot the computer, add "nameserver 8.8.8.8" to /etc/resolv.conf (which is the google nameserver)

Starting the cameras:

This morning, the cameras were started fine, upon fresh boot, without using coriander. So, the next section can be ignored, unless you have problems.
To start darc, as root:
darccontrol /rtc/conf/configfirewire.py --prefix=main
darccontrol /rtc/conf/configfirewire.py --prefix=sci

Actually, now, there are 2 new files:
darccontrol /home/dani/git/condor/configscao1.py --prefix=main
darccontrol /home/dani/git/condor/configsci.py --prefix=sci

You can also add a "-o" flag, which causes darc to output its output to the terminal. This can be useful while debugging, but prevents the logging from working.

I would also strongly recommend shutting down the computer each night. This way, people will get used to bringing up the system, and find ways to automate it. Otherwise, if it only reboots infrequently, you may be stuck trying to remember what to do.

Problems starting cameras?

If the cameras don't start properly when using darc, then start coriander (probably from a command line, since the launcher doesn't seem to know about it!).

This should present you with a list of 2 cameras, the guppy pro, and the unibrain fire-i.
First, select the unibrain, then in the display tab, set data rate to 400Mb/s (if seems to set at 100 by default), and then click display. Assuming the image looks okay, then quit coriander. If its not okay, then rebooting the computer may fix it!

Now it is time to start darc.

As root:
darccontrol /rtc/conf/configfirewire.py --prefix=main
darccontrol /rtc/conf/configfirewire.py --prefix=sci

This will start the 2 instances of darc. It is necessary to start as root, because I can't get permissions for USB access for the boston DM when as a normal user.

Status checking

Doing:
darcmagic status --prefix=main
(and also for --prefix=sci)
should tell you that they are both running, at about 15-30Hz.

darcplot --prefix=main
will bring up a real-time display (RTD). Select plotRawPxls.xml option.
Similarly for --prefix=sci

If you are having problems, try rebooting (or power cycling). The firewire bus seems to get into a state occassionally, and I've not yet figured out when.

Sorry its not very smooth! I expect its probably a simple configuration option that I'm missing!

The tool

In /home/dani/git/condor/, run tools.py. Looking at the code for this can also be instructive as you will learn how to code other scripts etc.

Configuration files

At the moment, I have only used one darc configuration file. However, I recommend that you set up lots... for all the different ways in which you intend to run the test bench. e.g. one for SCAO, one for the science camera, one for the Unibrain, one for when you get the new WFS, etc.
Also see git/condor/config*.py

Operating the DM

There are three possible DM configurations.

1. Boston DM only
2. Adaptica DM only
3. Both DMs

A separate darc module has been built for each configuration, allowing all configurations to be used.
These are:
libmirrorBMMMulti.so - Boston 160 actuator DM.
libmirrorAdaptica64.so - adaptica 48 actuator DM
libmirrorBMMMultiAdaptica64.so - Both, with adaptica having actuators 0-47, boston from 48-207.

To change, either change mirrorName in the configuration file, or use:
darcmagic set mirrorName -string=libmirrorWhatever.so --prefix=PREFIX
Then (if it isn't already open - which it will be if you didn't explicity close it):
darcmagic set mirrorOpen 1 --prefix=PREFIX

Note - because the adaptica driver was supplied as a 32 bit binary, it has been necessary to use a separate process with which darc communicates, to actually drive the mirror. This is started automagically when you load an adaptica mirror module, and is called mirrorAdapticaProcess (found in /rtc/bin). It will create a fifo at /tmp/adaptica.

Scripting

Use python.
```
import darc
d=darc.Control("main")#or "sci"

All sorts of things can then be done.
d.Set("dani",39.5)
bg=d.Get("bgImage")
stream=d.GetStream("rtcPxlBuf")#a single frame
data=d.GetStreamBlock("rtcPxlBuf",100)#100 frames - as a list
arrdata=numpy.array([x[0] for x in data])

#To sum some data:
data=d.SumData("rtcCentBuf",100)[0]
#and to get the mean:
data/=100

#To take a background:
bg=d.SumData("rtcPxlBuf",100,"f")[0]/100
```

```
d.Set("bgImage",bg)
#Note, you could also create a background using the camera calibration tool.

#Create ref slopes.
ref=d.SumData("rtcCentBuf",100)[0]/100
d.Set("refCentroids",ref)

#To do some poking...:
nacts=d.Get("nacts")
nslopes=d.Get("subapFlag").sum()*2
d.Set("addActuators",0)
#open the mirror library (if not already)
d.Set("mirrorOpen",1)
#Allow darc to talk to the DM (sorry - the naming isn't quite intuitive here).
d.Set("closeLoop",1)
pmx=numpy.zeros((nacts,nslopes),numpy.float32)
acts=numpy.ones((nacts,),numpy.float32)*32768
r=5000
n=10
#Note, at the moment, this assumes a linear response from the DM.  For the boston, this isn't
the case, since quadratic, so something needs to change in the boston darc module.
for i in range(nacts):
   acts[i]+=r
   d.Set("actuators",acts)
   time.sleep(0.05)#wait briefly
   c1=d.SumData("rtcCentBuf",n)[0]/n
   acts[i]-=r*2
   d.Set("actuators",acts)
   time.sleep(0.05)
   c2=d.SumData("rtcCentBuf",n)[0]/n
   acts[i]+=r
   pmx[i]=(c1-c2)/2./r
d.Set("actuators",acts)
#Now do least squares inversion
rmx=numpy.linalg.pinv(pmx,0.1).T #I don't think the -ve sign is required.

d.Set("rmx",rmx)
#set the gain
gain=d.Get("gain")
gain[:]=0.1
d.Set("gain",gain)

#and allow the loop to close
d.Set("addActuators",1)
```

#Note - may have got signs wrong in places... eg refCentroids, and rmx.

Firewire camera interface

Several parameters can be changed.

Set fwPrint to 1 (darcmagic set fwPrint 1 --prefix=main) to get a printout of values, ranges etc.
Then you can also set:
fwBrightness - no idea what this does
fwExposure - again, no idea
fwFrameRate - Sets framerate. 36 gives 30Hz, 35 gives 15Hz, 34 gives 7.5Hz etc.
fwGain - sets the gain.
fwPacketSize - not needed - useful in mode7.
fwShutter - not needed, but sets exposure time in mode7 I think.

Probably the only ones you would want to change would be fwGain and fwFrameRate. The others also seem to do something...
Note, these can be set in the parameter config file, or just set as needed from python/ darcmagic.

TODO

Insert a formula in the boston DM drivers to convert from linear to quadratic. To do this, have a look at /home/dani/git/darc/src/mirrorBMMMulti.c and also mirrorBMMMultiAdaptica64.c

You probably want to put it in the dmWrite method. If you do this, you should also change the values of actMin and actMax in the configuration file - these specify the range of allowed actuator values, but in this case would be the range allowed before conversion to quadratic, so that after conversion, the values don't exceed the 16 bit dac range...

Closing the loop

Take background, set it, then ref slopes, set them. Then do a poke. Invert the poke, and set as rmx. Also set the gain.

Then you need to set addActuators to 1, and set closeLoop to 1. You also need mirrorOpen==1 and closeLoop==1 before poking, otherwise it won't talk to the mirror.

DM safety

There is the facility to automatically open the loop if the DMs start clipping, as a safety feature. This is particularly useful if you are messing round with the system.
To use this feature:
set maxClipped to the maximum number of actuators that are allowed to clip before the loop opens. 10 is probably a good value.
Then set openLoopIfClip to 1.

If you do this, you will probably find that sometimes a display of rtcActuatorBuf freezes. This happens when the loop is opened (ie clipping occurs), and commands are no longer being sent to the DM.

To reset the system, check that rtcMirrorBuf looks okay (i.e. that reconstruction is working), set the gain low, and then set closeLoop to 1.

Plot configurations

Examples are in /home/dani/git/darc/conf/plot*.xml

Create your own, and place in /rtc/conf/. Alternatively, if you don't want to have any of the default plots, create your own plot directory, put all your plot configurations in it, and start a plotter with darcplot --prefix=main -c /path/to/your/dir/

Simplexing

Or removing non-common path errors.

The idea is to adjust the DM until the science image appears nicest. This should be done by adjusting the refCentroids.

Git repository

The repository is currently hosted in Durham, just to keep things backed up - however, I suggest you change this - all you need is a linux machine somewhere that you can log into.

To add new files to the repository (that already exist):
cd git/condor
git add filename

git commit -a
#then to push to the repository:
git push

To create a new repository:

ssh repository machine
mkdir git
cd git
git clone dani@darc.ing.puc.cl:git/condor --bare

You then have a machine hosting the repository. So, now you need to tell the darc machine about it.

On darc, cd git/condor.
git remote add --track master origin user@remoteRepositoryMachine:git/condor.git

You can now easily make a copy of the repository elsewhere...
ssh yetanothermachine
git clone user@remoteRepositoryMachine:git/condor.git

Now, whenever you create or modify files, you do
(git add filename if it is newly created)
git commit -a
git push

To update a repository if changes were made elsewhere...
git pull

LGS SHS images

python
data=FITS.Read("lgsImg.fits")[1]
datalgs=data[:,65536:]
datalgs.shape=100,128,128 # 100 frames of data.
datangs1=data[:,:128*256:1]
datangs1.shape=100,128,128
#and clip out a funny row...
datangs1=datangs1[:,:,:-1]

datangs2=data[:,1:128*256:1]
datangs2.shape=100,128,128
datangs2=datangs2[:,:,:-1]

```
datangs3=data[:,128*256::2]
datangs3.shape=100,128,128
datangs3=datangs3[:,:,:-1]

datangs4=data[:,128*256+1::2]
datangs4.shape=100,128,128
datangs4=datangs4[:,:,:-1]
```