Sony Pet Getting Started Guide

November 19, 2002

1 Getting access

1.1 Source code

How to check out a copy of the source code.

- Make a directory to put the code in and change to that directory

```
mkdir dogs
cd dogs
```

- Check out a copy of robot code (agent) and utility code (util)

```
export CVS_RSH=ssh
cvs -d :ext:gs104.sp.cs.cmu.edu:/data/dogs/code_repository \
  co agent util
```

- If you get a premission denied error, Scott probably needs to fix permissions

1.2 Sony documentation

- Load the Sony RoboCup support site
  https://www.openr.org/page1_2003/index.html
- Load the Sony OPENR-SDK support site
  https://www.jp.aibo.com/openr/e_regi/index.jsp

1.3 Our documentation

Where to access our publications and documentation (respectively)
[on gs104.sp.cs.cmu.edu]

```
/data/dogs/docs/papers/
/data/dogs/docs/doc/
```

2 Compiling

2.1 Setup

- Set DOGROOT (directory with code) variable

```
export DOGROOT=~/dogs
```

If you get an error which says 'export: command not found', see the appendix for how to change your shell to zsh or translate the command for your shell.

- Add to your PATH variable

```
export PATH=/usr/local/OPEN_R_SDK/bin:${DOGROOT}/util/bin:${PATH}
```

2.2 Compiling

How to compile the code.
[from dogs/agent directory]

```
make
```

If you get errors about unable to find mipsel-*, either your path is incorrect or the machine is set up wrong. Run 'echo $PATH' and make sure that /usr/local/OPEN_R_SDK/bin is in your path.
If you get errors about unable to find dep_process, your path is missing

```
${DOGROOT}/util/bin
```

3 Running

3.1 Setup

3.1.1 Selecting the behavior run

How to set up behavior configuration for a memory stick.

- Put the memory stick in a card reader.
- To set the behavior that you want to run you must access the stick, Either:
  - * `mount /memstick`
    `cd /memstick/config`

    * Edit behave.cfg. Enter desired behavior in the `""` after `role=`, and save file.
    * Stop the stick access by running
      `stopit`

      If you are on a machine using a USB memory stick reader (not a laptop), you can also use `umount /memstick` to safely stop the stick.
  - `beit <behavior name>`
- Remove the memory stick

3.1.2 Selecting the robot to use

To configure for the robot being used, run:

```
~/dogs/util/wlan_setup -i <ers0?>
```

Where the last argument is the machine name of the robot (one of `ers01`, `ers02`, `ers03`, `ers04`, `ers05`, `ers06`, `ers07`, `ers08`)

3.1.3 Copying the code to the stick

How to put your code on a memory stick.

- Put the memory stick in the card reader.
- Download your code to the memory stick.
  Three file options (you must specify at least one):
  - -a (everything, usually use this)
  - -b (binaries/executables)
  - -c (configuration files)

```
stickit -a
```

- Remove the memory stick from the reader. Stickit starts and stops the card reader drivers as necessary.

3.2 How to run the robots

- Put the memory stick and a battery in a dog
- Turn on the robot by pressing chest button and wait ∼10 seconds for dog to boot.
- To start robot action (if desired), press back button.
- Pause robot (if desired) by holding back button 1 second (push again to unpause). If you hold it too long, the robot will unpause again. You can tell when you have held it long enough when the tail twitches.
- Stop robot by pushing chest button (no way to undo) and wait for activity to cease.
- Take out memory stick, and robot will shut itself off.

3.3 Running the GUI

3.3.1 Setup

- Compile chokechain

```
cd ~/dogs/util/choke_chain
make
```

- Check to be sure OPENR_LINUX (not openr_linux) is in expected directory.

```
cd ~/dogs/util/RoboCup2002-05-14/snap.EachTeam
export OPENR_LINUX=/usr/local/OPENR_LINUX
$OPENR_LINUX/scripts/configure
make
```

- Setup robot you want to run by uncommenting it out in port.cfg file in two places using `xemacs` or another editor. Make sure you comment out any robots that you are not using.

3.3.2 Running the GUI

How to run the GUI.

- Run chokechain

  ```
  ~/dogs/util/choke_chain/chokechain
  ```

- In the window open the views you want (world or RLE)
- Boot the robot
- Run oobjectManager. You can kill oobjectManager and restart without restarting chokechain.

  ```
  ~/dogs/util/RoboCup2002-05-14/snap.EachTeam/oobjectManager
  ```

- Press return in the oobjectManger window to start it
- Unpause the robot
- Sometimes resources are not properly freed by oobjectManager. Run

  ```
  $OPENR_LINUX/scripts/ipcrmall
  ```

  to free these resources.

A Using UNIX

A.1 Basic commands

`ls` List the contents of a directory.

`cd <dir>` Change the current directory.

`pwd` Print the current directory.

`less <file>` View a text file.

`xemacs <file> &` Start the xemacs editor in a separate window on a file.

`xemacs -nw <file>` Start the xemacs editor in the current window on a file.

`man <prog name>` Get help on a program, output will come up in a pager.

`rm <file name>` Remove a file.

A.2 Using a pager

Here is a list of basic commands to use the common pagers `less` and `more`.

`<space>` Advance to next screen full of information.

`b` Go back one screen full of information.

`q` Quit pager and return to shell.

`<return>` Advance one line.

`/<regular expression>` Search for next instance of the regular expression. You can search for strings containing only letters, numbers and underscores simply by using the string as the regular expression.

A.3 Using a shell

These instructions assume that you are using zsh. If you want to use another inferior shell, you are on your own. You can check what shell you are currently running by running `ps`. The item it lists that ends with "sh" is the shell you are currently using.

To change your shell, run `chsh`. Type in your password when prompted. When prompted for the new shell, enter `/usr/bin/zsh`.

To set an environment variable, the command is `export <var-name>=<value>`. This sets the value for the current shell.

To set an environment variable for all new shells that you create, add the line `export <var-name>=<value>` to the file ∼/.zshrc

To avoid typing the same command over again, you can use up and down arrow to cycle through the previous commands you have entered.

You can type `<tab>` in many situations at the shell to complete things for you or get a list of possible completions in case it is ambiguous.

A.3.1 UNIX paths

Unix paths all start at the root directory (`/`). The current directory is indicated by period (`.`). The parent of the current directory is indicated by two periods (`..`). The components of a path are separated by forward slashes (`/`).

A.4 Using xemacs

Type text using the keyboard. Use the arrow keys and such to move around.

Type `C-x C-s` to save the current file (or use the button or menu options). `C-x` means press the "Ctrl" key, press the "x" key, release the "x" key, then release the "Ctrl" key.

Type `C-x C-c` to quit xemacs.

A.5 Using CVS

A.5.1 Setup

Add the following lines to your `/.cvsrc` file.

```
update -dP
checkout -P
```

A.5.2 CVS commands

`checkout <module>` or `co <module>` Checks out the code module specified. Use this to get an initial copy (called a working copy) of a piece of code.

`update` Updates the working copy recursing from the current directory to incorporate the latest changes in the repository. CVS will attempt to merge the changes in the repository with whatever local changes you have. CVS will display a "C" next to filenames that had conflicts. You will have to manually edit these files to resolve the conflicts. Search for "===" to find the conflict markers.

`diff` Displays differences between files under CVS control. Without options, displays all changes you have made locally to the version you checked out (**Note: this will not display anything about changes other people have made to the repository since you checked out.**). This command can also show differences between your working copy and any version in the repository and between any two revisions in the repository.

`commit <filename1> <filename2>` Commits the files specified to the repository where other people can get the changes using `update`. If no files are specified, `.` is used. Directories are recursed to pick up changes to be committed. If you have not updated to the latest version of the repository before you try to commit, CVS will print out an error and abort. CVS will bring up an editor after determining what files need committed to allow

you to enter a log message. Type a message in the editor describing the changes you made and then save the file and exit the editor. Make sure to use a descriptive commit message.

`log <filename>` Displays the commit messages that have been entered for this file which describe the changes done. Also displays some summary statistics about the changes performed on the file.

`add <filename>` Add file or directory to the repository. Files are added when they are committed, directories are added immediately. Please be sure of the need for a new directory before adding a new directory to the repository.

`remove -f <filename>` Remove the file from the working copy and schedule the file for deletion in the repository. **There is no way to recover the local file after performing this operation.** There is no way to remove directories. If the `-P` option is passed to `update/co`, empty directories in the repository will be removed from the working copy on update. Without the `-f` option, the local file must already be removed and the command schedules the file for deletion in the repository.

A.5.3 Common CVS options

Here are some options used before the CVS command:

`-n` Fake running the command. Generate all output as normal but don't actually do anything.

`-d <repository>` Specify a repository to use when performing the CVS command. This option is usually not needed because CVS defaults to using the repository location stored in the working copy.

Here are some common options used by CVS commands:

`-r <rev>` Used to specify a revision.

`-D <date>` Used to specify a date for a revision.

`-d` Specifies that `update` should add directories if necessary.

`-P` Specifies that `update` and `checkout` should prune empty directories.

A.5.4 CVS environment variables

Here are some environment variables used by CVS:

`CVS_RSH` By default CVS uses the insecure protocol rsh to connect to `:ext:` repositories. Setting this to `ssh` makes CVS use the ssh protocol.

`CVSROOT` Specifies the location of the repository. This variable usually doesn't need to be set since it is determined automatically from the working copy.

A.5.5 How to perform common tasks

Here are some common tasks that are done and the way to do them using CVS:

Creating a working copy `cvs -d <repository loc> co <module name>`

Finding out what changes other people have made `cvs -n update`. This lists the effect that an update command will have on the working copy. The letters in front of the file names mean:

- **`M`** Modified locally.
- **`U`** Modified in repository.
- **`P`** Modified in repository.
- **`C`** Modified locally and in repository.

? Local file not present in repository.

A Locally added file.

D Locally deleted file.

Find out what changes you have made. Use `cvs -n update` to find any files you have added/deleted that need to be scheduled for addition/deletion from the repository. Use `cvs diff` to determine what changes you have made to each file.

Move a file. Use `mv <old-name> <new-name>` to move the file. Use `cvs remove <old-name>` to remove the old file name from the repository. Use `cvs add <new-name>` to add the new file name to the repository. Include a note in the commit message mentioning that the file was moved and the new and old names.

Remove a directory. Remove all of the files in the directory. CVS does not directly version directories.

Commit changes. Use `cvs commit <filename1> <filename2>` to commit your changes. The filenames can be directories in which case they will be recursed upon to look for changes. You can omit the names to indicate the current directory.

A.6 Using ssh/scp

We do not use `telnet` or `ftp` on our machines because they are insecure protocols (unless using kerberized version) which send passwords and other sensitive information in cleartext. Instead, we use the ssh tools `ssh` and `scp`.

`ssh` is a secure replacement for `telnet` that also securely forwards X11 requests. This allows access to remote shells and the ability to run X11 programs on the remote machine with the display sent to the local machine. To connect to a remote machine use `ssh [<user>@]<machine name>`. The user name can be omitted if it the same on both machines. The brackets indicate an optional part of the command.

`scp` is a secure alternative to `ftp`. `scp` uses a syntax similar to `cp`. To specify a remote file use the syntax `[<user>@]<machine>:<filename>`. The user name can be omitted if it is the same on both machines. The filename is a path relative to your home directory unless it starts with a slash (in which case it is relative to the root directory). The rest of the syntax is the same as `cp`. See `man scp` for more details.

A.6.1 Using public/private key authentication

Normally `ssh` and `scp` authentic by asking for your password. You can set up ssh to use public/private key authentication instead. This creates a trust relationship between the machines, so please only use this between machines within the lab. Here are the steps needed to setup public/private key authentication so that remote machine trusts the local machine:

1. Open two xterms.
2. Connect to the remote machine using `ssh` in one window.
3. Run `ssh-keygen -t dsa -b 2048` on the local machine (you only have to do this once per machine). Just hit enter when it asks for a pass phrase.
4. On the remote machine run `cd ~/.ssh`.
5. On the local machine run `cat ~/.ssh/id_dsa.pub`.
6. On the local machine select the print out with the mouse. The text should be completely highlighted starting with “ssh-dsa” through the end of the line containing your login name and the machine name.
7. On the remote machine run `cat - >>~/.ssh/authorized_keys`.
8. With the remote window focussed, press the middle mouse button to paste the ssh key.
9. Type `C-d`, i.e. press “Ctrl”, press “d”, release “d”, release “Ctrl”.
10. You should now be able to ssh to the remote machine from the local machine without entering a password.