Installing Player and Stage in Ubuntu

From Robots in everyday human environments

This installation guide has been tested on *Ubuntu* (http://www.ubuntu.com/) (version 9.04 - Jaunty Jackalope), but should work on other Linux distributions as well.

Before installing *Player* (http://playerstage.sourceforge.net/index.php?src=player) and *Stage* (http://playerstage.sourceforge.net/index.php?src=stage) in *Ubuntu* (http://www.ubuntu.com/) , certain dependency packages must be installed first. The packages are listed in the box to the right, and can be installed either from the graphical installer *Synaptic* (http://help.ubuntu.com/community/SynapticHowto) or from the terminal using *apt-get* (http://help.ubuntu.com/community/AptGetHowto) or *aptitude* (http://help.ubuntu.com/community/AptitudeSurvivalGuide) .

Package name
autotools-dev
build-essential
cmake
cpp
libboost-thread1.35.0
libboost-thread1.35-dev
libboost-signals1.35.0
libboost-signals1.35-dev
libcv1
libcv-dev
libgdk-pixbuf2
libgdk-pixbuf-dev
libgnomecanvas2-0
libgnomecanvas2-dev
libgsl0
libgsl0-dev
libjpeg62-dev
libtool
libxmu-dev
swig
freeglut3 (for stage)
freeglut3-dev (for stage)
libfltk1.1 (for stage)
libfltk1.1-dev (for stage)
libgtk2.0-dev (for stage)
libltdl7 (for stage)
libltdl7-dev (for stage)
libpng12-0 (for stage)
libpng12-0-dev (for stage)

The list of dependencies was sufficient for the installation on the given Ubuntu version. But it might not be up to date with the newest versions, or others might not be necessary. Some drivers, which we do not use, were not installed due to missing dependencies. The list of drives not installed comes out when using cmake (see below). Note that you might need to restart after installing all the dependencies.

A script that installs the dependencies, downloads the installation files, and installs *Player* (http://playerstage.sourceforge.net/index.php?src=player) and *Stage* (http://playerstage.sourceforge.net/index.php?src=stage) has been created. The script can be found *here* (http://www.control.aau.dk/~ms/playerstage/playerstage_install) .

Contents

Player

Having downloaded and installed the above packages, *Player* (http://playerstage.sourceforge.net/index.php?src=player) must be downloaded. Currently (Oct. 09) the newest version is 3.0.0 and can be downloaded from *this site* (http://sourceforge.net/projects/playerstage/files/Player) .

First unpack the downloaded package, by using e.g. tar:

```
$ tar xzvf player-<version>.tar.gz
```

From *Player* (http://playerstage.sourceforge.net/index.php?src=player) version 3.0.0 a new build system has been

implemented. Instead of using a configure script, cmake is used to build the installation (see notes *here* (http://softlayer.dl.sourceforge.net/project/playerstage/Player/3.0.0/player-changelog-3.0.0)). This also make it possible to compile Player for Windows

Before version 3.0.0, the installation was configured like this:

```
$ ./configure --prefix=<INSTALL_DIR> --disable-alldrivers --enable-cmvision --enable-camerav4l
--enable-urglaser --enable-amcl --enable-vfh --enable-wavefront --enable-logfile --enable-mapfil
--enable-mapscale --enable-vmapfile --enable-bumpersafe --enable-lasersafe
```

where `<INSTALL_DIR>` is the destination path, which for example can be `/home/$USER/playerstage`. The default installation path (used when the `--prefix` command is omitted) is `/usr/local`, and it is therefore necessary to become root to use this installation directory. The reason for disabling all drivers, and enabling those needed is to reduce compilation time and the amount of required space. To obtain a complete installation of *Player* (http://playerstage.sourceforge.net/index.php?src=player) , omit all flags other than the `--prefix`.

To use default configuration settings in Player 3.0.0 and later use cmake with the path to the unpacked files. But first create a temporary build directory:

```
$ cd player-<version>
$ mkdir build
$ cd build
$ cmake ../
```

If not wanting to use default path, it is possible to use:

```
$ cmake -DCMAKE_INSTALL_PREFIX=<INSTALL_DIR> ../
```

For special configurations use

```
$ ccmake ../
```

When the configuration completes, and without errors, complete the installation with:

```
$ make
$ sudo make install   # You need administrator privileges if you install in the default /usr/loc
```

When *Player* (http://playerstage.sourceforge.net/index.php?src=player) , is installed, you can delete the unpacked and compiled installation files again.

When upgrading to Ubuntu 9.10, you might get an error when running *Player* (http://playerstage.sourceforge.net/index.php?src=player) . The error is something like:

```
 error while loading shared libraries: libgeos-3.0.0.so
```

This is because a new version of libgeos is used in Ubuntu 9.10. You can make a fix by making a symbolic link from to old to the new version:

```
$ cd /usr/lib
```

```
$ sudo ln -s libgeos-3.1.0.so libgeos-3.0.0.so
```

Stage

The currently (Oct. 09) most recent version of *Stage* (http://playerstage.sourceforge.net/index.php?src=stage) is 3.2.0 It can be downloaded from *this site* (http://sourceforge.net/projects/playerstage/files/Stage) . The official installation guide is *here* (http://playerstage.sourceforge.net/doc/stage-3.1.0/install.html) .

- If you plan to use Player with Stage, make sure *Player* (http://playerstage.sourceforge.net/index.php?src=player) is installed and working, otherwise it will not work. See the *Player* (http://playerstage.sourceforge.net/index.php?src=player) documentation for instructions.
- Uncompress and expand the tarball:

```
$ tar xzvf Stage-<version>-Source.tar.gz
```

- `cd' into Stage's source directory:

```
$ cd Stage-<version>-Source
```

- Configure *Stage* (http://playerstage.sourceforge.net/index.php?src=stage) with default settings. As with *Player* (http://playerstage.sourceforge.net/index.php?src=player) , cmake is now used to build stage. For earlier versions do:

```
$ ./configure
```

With cmake:

```
$ mkdir build
$ cd build
$ cmake ../
```

- Compile and install *Stage* (http://playerstage.sourceforge.net/index.php?src=stage) :

```
$ make
$ sudo make install
```

- As with *Player* (http://playerstage.sourceforge.net/index.php?src=player) , the default installation directory for <INSTALL_DIR> is `/usr/local`, so you need to become root for this step if using the default installation directory.

The stage installation does not copy all the example worlds, so if you want to keep them, copy them to somewhere else like e.g.

```
$  mkdir /home/$USER/playerstage
$  cp -r worlds/ /home/$USER/playerstage
```

When *Stage* (http://playerstage.sourceforge.net/index.php?src=stage) is installed, you can delete the unpacked installation files again.

Verifying the installation

First ensure that player is in the PATH, by typing:

```
$ which player
```

This should result in the following output:

```
<INSTALL_DIR>/bin/player    (e.g. /usr/local/bin/player  or /home/$USER/playerstage/bin/pla
```

If so, skip the first of the following two lines. Add the these lines to your `.bashrc` file located in the root of your home directory:

```
export PATH="$PATH:<INSTALL_DIR>/share/player/bin"
export LD_LIBRARY_PATH=<INSTALL_DIR>/lib
export STAGEPATH=<INSTALL_DIR>/lib
export PLAYERPATH=<INSTALL_DIR>/lib
```

Run the `.bashrc` file by:

```
$ source .bashrc
```

Now everything should be working. To test *Stage* (http://playerstage.sourceforge.net/index.php?src=stage) , move to the installation directory and issue the following command in a terminal window:

```
$ cd <STAGE_WORLDS_INSTALL_DIR>
$ stage simple.world
```

If you encounter any error saying something like (only in earlier versions of Stage/Ubuntu)

```
err: unable to open color database /usr/X11R6/lib/X11/rgb.txt : No such file or directory (
```

...it is most certainly caused by the fact that the X11 color description file is placed in another directory, probably `/usr/lib/X11`. If so issue the following command:

```
$ ln -s /usr/lib/X11/rgb.txt /usr/X11R6/lib/X11/rgb.txt
```

Then try to run the `stage` command again. If the `rgb.txt` file is still causing trouble, try to copy it instead of linking.

If the `player` command did not cause any errors, we are ready to test the ability to use *Stage* (http://playerstage.sourceforge.net/index.php?src=stage) plugins with *Player* (http://playerstage.sourceforge.net/index.php?src=player) . Issue the following commands:

```
$ cd <STAGE_WORLDS_INSTALL_DIR>
$ stage fasr.world
```

You can also try to make the simulation from player by running

```
$ player simple.cfg
```

Doing this you might end up with the problem that the *Stage* (http://playerstage.sourceforge.net/index.php?src=stage) GUI freezes. The solution to this is to go back to the *Stage* (http://playerstage.sourceforge.net/index.php?src=stage) source code. Open the file libstageplugin/p_driver.cc, and change the line:

```
world->Update();
```

to

```
Fl::wait();
```

Then recompile and install using make and make install like described above. I guess that this will be fixed in future versions of *Stage* (http://playerstage.sourceforge.net/index.php?src=stage) .

Controlling your own robot manually

Now you can try to add a robot, which you can control yourself. First add a new robot to the world by appending the following lines to `simple.world`:

```
pioneer2dx
(
  name "myrobot
  color "green"
  pose [ 0 3 0 0 ]
  sicklaser()
)
```

Then append the following lines to the *Player* (http://playerstage.sourceforge.net/index.php?src=player) config file `simple.cfg`:

```
driver
(
  name "stage"
  provides [ "position2d:1" "laser:1" ]
  model "myrobot"
)
```

Then start up the simulation by running

```
$ player simple.cfg
```

You should now see a simulation with two robots. A red one is wandering around like before, and a green one, which is not moving. To manually control the green robot, open a new terminal window and type:

```
$ playerv
```

Go to the click on "Devices" -> "position2d:1 (stage)" -> "Subscribe", afterwards "Devices" -> "position2d:1 (stage)" -> "Control", and "Devices" -> "laser:1 (stage)" -> "Subscribe". Then, by moving the red square, you should be able to control the robot in the *Stage* (http://playerstage.sourceforge.net/index.php?src=stage) window.

Compiling a control program

Player (http://playerstage.sourceforge.net/index.php?src=player) comes with some example code for controlling a robot. If you installed in the default location it will be in:

```
$ /usr/local/share/player/examples/
```

This directory is not writable, unless you are root, so it is recommended to copy it somewhere else, if you want to use the examples.

In the subfolder `libplayerc++/` there is a file called `libplayerc++/`. This example is compiled like this:

```
$ g++ -o laserobstacleavoid `pkg-config --cflags playerc++` laserobstacleavoid.cc `pkg-config
```

You can avoid the first pkg-config by exporting a CPATH, or setting it in your `~/.bashrc` file:

```
$ export CPATH="$CPATH:/usr/local/include/player-3.0"   (if Player is installed in the default
```

and compile:

```
$ g++ -o laserobstacleavoid  laserobstacleavoid.cc `pkg-config --libs playerc++`
```

If you did not install at the default location, you might also have to export the PKG_CONFIG_PATH:

```
$ export PKG_CONFIG_PATH="<INSTALL_DIR>/lib/pkgconfig/:$PKG_CONFIG_PATH"
```

To test if it works, you can use the above modified version of the `simple` world.

```
$ player simple.cfg
$ ./laserobstacleavoid -i 1  (in a new terminal)
```

If you just type `./laserobstacleavoid` (without `-i 1`), you will control the default robot (number 0). This robot is already set to "wander" in the `simple.world`, and you will therefore not see any effect.

Now, to control a real robot, you "just" have to run a player server on the robot, instead of controlling the *Stage* (http://playerstage.sourceforge.net/index.php?src=stage) simulation :-)