A Collaborative guide to ATLAS Development

R. Clint Whaley *

September 12, 2001

Abstract

This paper exists to get you started if you want to do some ATLAS development. The hope is that as new developers work on ATLAS, they will add to this note, so that this information grows with time.

*Dept. of Computer Sciences, Univ. of TN, Knoxville, TN 37996, `rwhaley@cs.utk.edu`

Contents

1 Introduction **3**

2 Adding to this note **3**

3 Getting the ATLAS code through CVS **4**

3.1 Background on ATLAS code base 4
3.2 Getting the ATLAS CVS tree and the working ATLAS directory 4
3.3 Basefile/extracted file interaction 5

1 Introduction

This note is exists to get you started if you want to write **new** code for ATLAS, or if you want to modify ATLAS source. It is *not* for kernel contribution, which is what most people do when contributing to ATLAS. Kernel contribution is much simpler, and is explained in `ATLAS/doc/atlas_contrib.ps` or in html format at:

`http://math-atlas.sourceforge.net/atlas_contrib`

So what is the difference development and kernel contribution? In kernel contribution, you write a kernel to be used by ATLAS, using the provided ATLAS testers and timers to verify it, and when you are satisfied with it's performance and reliability, you submit your kernel to the ATLAS team, and they accept it or not, and you are done.

Doing actual core development is quite a bit more complex. Probably the biggest change is that you will need to write your own tester and timer for your new code. No code will be accepted into the ATLAS code base without a tester which can be used to verify it. Since writing a decent tester is usually at least as hard as writing the code it tests, and is always a whole lot less enjoyable, the author must bear the pain of producing it along with the pride of producing the code. As a developer, you will be responsible for testing such new code on several platforms as well.

If you are instead hoping to modify some of the existing code base, remember that for non-kernel operations, portability and robustness *must* be the primary goal. There are many sections of ATLAS that we know to be second rate on a certain platform, but we also know that it works on the twenty or so architectures that ATLAS is routinely compiled for, so we leave it that way. This means that when a modification is made to a previously existing routine, the modifying author must have good evidence that the new code is as portable as the old. In short, the barrier to replacing tested code is high.

It is possible that users want access to the CVS repository even though they do not plan on doing development, mainly I'd guess so they can get access to the newest stuff without waiting for developer releases. Also, kernel contributers who make subsequent changes to their routines can speed up their adoption by submitting them in a format ready for CVS checkin.

2 Adding to this note

This note is included in the `AtlasBase/TexDoc` of the CVS repository, and anyone can submit a patch against it giving additional information. As the founder of ATLAS, I have written a seed of a document explaining how to get access to the code. It is my hope that other developers will add important information that they discover as they go, so that this doc will grow over time, getting much information that I probably take so much for granted that I would never think to document.

Adding section makes a lot of sense, and probably a FAQ appendix would be a good idea. As people contribute, their names will be added to the author list.

3 Getting the ATLAS code through CVS

3.1 Background on ATLAS code base

ATLAS was not originally developed in CVS. ATLAS was developed using a programming tool called *extract*, which means ATLAS is actually maintained in something called basefiles. If you think of regular development being access by value, and CVS using a level of indirection, CVS on basefiles gives you two such levels of indirection. So, if you want to be able to use CVS checkins, you will need to learn at least the basics of extract. Details on extract can be found at:

```
http://www.cs.utk.edu/~rwhaley/extract/
```

Note that if you just want read-only access, you will need to install extract so that you can get at the files, but you will not need to learn anything about it.

3.2 Getting the ATLAS CVS tree and the working ATLAS directory

Here are the steps to get a local ATLAS CVS tree:

1. **Install extract**: Download and install extract as described here:

   ```
   http://www.cs.utk.edu/~rwhaley/EXTRACT/faq.html#install
   ```

2. **Checkout the ATLAS CVS tree**: From a CVS-capable machine connected to the internet, go to the directory where you want the ATLAS *basefiles* to be, and issue:

   ```
   cvs -z3 -d:pserver:anonymous@cvs.math-atlas.sourceforge.net:/cvsroot/math-atl
   as checkout AtlasBase
   ```

3. **Update `topd`**: Edit the created `AtlasBase/make.base`, and change the definition of the `topd` macro to match where you have put the AtlasBase directory. For example, if I issued the CVS command in `/home/rwhaley/work`, I change line 46 from something like:

   ```
   @define topd @/home/rwhaley/Base/AtlasBase@
   ```

   to:

   ```
   @define topd @/home/rwhaley/work/AtlasBase@
   ```

4. **Create ATLAS working directory and extract files**: Go to the directory where you want the working copy of the `ATLAS/` directory to reside. In there, issue these two commands:

   ```
   extract -b <your topd>/make.base -o Makefile rout=Make.atldir
   make
   ```

3.3 Basefile/extracted file interaction

You'll have to scope the extract page for any kind of real feel for how this works, but some atlas-specific details are in order here. In each subdirectory of the `ATLAS/` tree, you will find a file `Make.ext`. If you type `make -f Make.ext`, this makefile will extract all new files in this subdirectory for which the basefile is newer.

So, what usually happens is you a messing with something, and you do it in the `ATLAS/` directory. When you are confident in your change, you put it into the appropriate basefile in the AtlasBase/ directory (note that examining `Make.ext` will show you what basefile a given extracted file comes from), and you then re-extract over your working copy with the above command.