Forget **make**, here is:

A-A-P handout

FOSDEM 2003

February 8 - 9

www.A-A-P.org

The A-A-P project has four **goals**:

Develop	Make it easier to develop software together with others connected by the Internet. A framework of powerful tools is provided, items are automatically downloaded when needed.
Distribute	Support automatically uploading changed files to a web server or committing to a CVS server.
Find	Help a user to find software. On a central site software developers and users can exchange recipes for installing an application on any system.
Install	Make it simple to install software. A user only needs to get a short recipe file. When it is executed his system is inspected and the required files are downloaded, patched, compiled and installed.

The main parts of A-A-P are:

Aap	A program that executes recipes. It can do what "make" does, plus working with remote files, supports signatures and much more. A beta version is available. **Try it out!**
Agide	An IDE framework where separate tools are bound together to work productively. Use the editor of your choice, search for items in a project, design a dialog, etc. Agide is currently being developed, the first usable version is expected in March 2003.

Recipe example: **compile a C program**

```
SOURCE = main.c
         general.c
         util.c
TARGET = tryout
```

Dependencies on header files will be detected automatically.
MD5 signatures are used to detect changed files.

A **recipe** is a clever combination of the simplicity of a Makefile with the power of Python. Dependencies are specified like in a Makefile. But the need for all those backslashes has been removed, line continuation is indicated by indent.
Using a signature instead of a timestamp avoids problems when an older version of a file has been restored and when the clock of two machines does not match.

Recipe example: **publish a web site**

```
FILES  = index.html header.html about.html
         `glob(“images/*.png”)`

:attr {publish = scp://user@ftp.foo.org/html}
                          $FILES
all : publish
```

Specify the list of files to be uploaded and the URL of the server.
The Python function “glob” is used to find all image files. Backticks enclose a Python expression.
A file will only be uploaded when its MD5 signature has changed.

Python script can be used in a recipe where needed. This provides a portable scripting language, making it possible to write recipes that execute both on Unix and MS-Windows. Shell commands can still be used, but do make a recipe system specific.

Recipe example: **generate a file**

```
version.h : version.h.in

    :cat $source | :eval re.sub(‘@date@’,
                       DATESTR, stdin) >! $target
    :print Changed date in $target to “$DATESTR”.
```

The :cat command works like the Unix cat command.
The :eval command can be used in a pipe to filter text through a Python function.

Several built-in commands are provided to make it possible to write portable recipes. They mostly work like Unix shell commands. There are extra features, such as support for handling remote files by their URL.

Recipe example: **using CVS**

```
VERSION = 1.013

CVSROOT = :ext:$CVSUSER@cvs.foo.org:/cvsroot/foo

FILES = COPYING README.txt
        `glob("*.py")` main.aap

:attr {commit = cvs://$CVSROOT}
      {logentry = update for version $VERSION}
         $FILES
```

CVSROOT specifies the location of the CVS server. CVSUSER is the user name that must be specified.
The "commit" attribute specifies what version control system is to be used and where to find it.
The "logentry" attribute can be used to specify a message for an updated file.

Updating to the latest version is done with the command "aap fetch". Committing local changes is done with "aap revise". Aap will automatically add new files and remove deleted files. Note that using "*.py" to select files is actually a bit dangerous, a Python file may be accidentally checked in.

More examples and documentation can be found on the web site.

Warning: The programs have not been thoroughly tested yet, USE WITH GREAT CARE!

How can you help A-A-P?

A-A-P is an open-source, free software project using the GNU GPL. You are invited to join in and help making this project a success.
Bram Moolenaar is currently doing much of the work, it is his full-time job. This makes sure the project will at least provide the basic functionality. But he can only do a limited amount of work, while there are very many good ideas. To make A-A-P a success we need help!

Here are some of the tasks that are available:

- **Add support for a language**
  This involves providing the default rules for the language. Possibly a dependency checker can be added, so that dependencies on included files are handled automatically. Most of this can be done with recipe commands. For some things Python may need to be used. SCons can be used as an example.
- **Add support for a compiler**
  Especially on MS-Windows each compiler requires specific options. This work involves providing a check for which compiler is going to be used and setting the defaults. SCons can be used as an example.
- **Continue work on the port recipe**
  The port recipe provides a portable way to compile and install a package on BSD systems. This was only partly implemented because of lack of time. This is to be implemented in Python.
- **Documentation**
  Most of the documentation is currently in plain text. This should be converted to DocBook format. Add links to be able to jump around the manual. Add an index.
- **Example projects**
  Write recipes for a whole project and present it to others. This can function as an example of how A-A-P is used. It is also a check for the functionality and can be used as a "success story".
- **Testing**
  Python is a nice language, but it doesn't do type checking. A lot of testing is required to avoid bugs. Write random recipes to find problems. Write specific test recipes and unit tests for the Python code, to be included with the test suite.
- **Create a web site for recipe exchange**
  Set up a database and a web site where developers can upload their recipes and users can download them. The Vim online site can function as an example.
- **Add support for a transport method**
  Currently transport through http, ftp and scp is supported. It would be nice to have more methods.
- **Add support for a version control system**
  Currently only CVS is supported. Would be nice to have support for SCCS, RCS, Subversion and others.
- **Add support for an issue tracker**
  Make it easy to enter a bug report or feature request for specific bug tracking systems.
- **Automatic configuration**
  Help creating a portable autoconf replacement. By implementing this in Python it will work on non-Unix systems and work much simpler.

Contact Bram Moolenaar if you have questions: <Bram@a-a-p.org>