

# aCe C User's Guide

John E. Dorband  
*NASA Goddard Space Flight Center*  
*Greenbelt, MD 20771*

## **Introduction**

aCe stands for architecture-adaptive computing environment. aCe C is a superset of ANS C. In this document, the term aCe will be used synonymously for aCe C. This description of aCe C assumes that the reader is knowledgeable of ANS C. This document shows how to install aCe, compile an aCe program, and run an aCe executable.

Currently, aCe runs on single and multiple Linux processors connected.

## **Installation**

Documentation for aCe can be found in the './share/ace' directory (relative to the 'bin' directory in which ace is found). aCe includes no third party software. However if one wants to use aCe graphics they must install either 'glut' or 'freeglut'. Otherwise glut is not necessary.

Installation (for LINUX):

- Put ace-1.00.xx.tar.gz in '/usr/local'
- If aCe graphics is to be used, glut or freeglut  
must be installed (Fedora Core includes freeglut).
- De-tar aCe: 'tar xzpf ace-1.00.xx.tar.gz' in '/usr/local'
- Goto aCe directory: 'cd ace-1.00.xx '
- Run INSTALL: ' ./INSTALL '
- It is not necessary but the installation should  
be checked: 'make check '

### Installation Prep (for MAC OsX 10.3):

- Install X11 for Mac:  
X11User.pkg on disk 3 of installation disks.
- install programming development environment:  
Developer.mpkg from Xcode Tools disk
- Install X11 sdk:  
X11SDK.pkg from Xcode Tools disk
- Install Debian Mac software (Fink):  
from <http://fink.sourceforge.net/>
- Once Fink is installed, use 'FinkCommander' to  
install 'glut' from source code. use the button  
to install from source, NOT from binaries.

### Installation (for MAC OsX):

- Put ace-1.00.xx.tar.gz in '/sw/src'
- De-tar aCe: 'tar xzpf ace-1.00.xx.tar.gz' in '/sw/src'
- Goto aCe directory: 'cd ace-1.00.xx '
- Run INSTALL.macosx: ' ./INSTALL.macosx '
- It is not necessary but the installation should  
be checked: 'make check '

## Compilation

An aCe program is compiled very similarly to C programs. The command is 'acecc'. aCe sources files have a suffix of '.aCe'. aCe include files have a suffix of '.aHr'. An aCe source file is compiled into an aCe link or object file, and its suffix is '.aLk'. Besides aCe source and object file names, the 'acecc' command line may also take standard '.c' and '.o' file names. A typical aCe compile command might be:

```
acecc HelloWorld.aCe
```

This command will produce a standard 'a.out' file that may be executed like any other Linux executable. (Note that this is only a single-process executable.) To name the executable something different than 'a.out', the following command will create an executable with the name 'HWorld':

```
acecc HelloWorld.aCe -o HWorld
```

There are two ways to tell the aCe compiler to create a multi-process executable. The first is by designating the type of the executable. A type of 'skt' indicates an

socket-based executable. A multi-process executable can only be executed with the ‘acerun’ command. This will be discussed in the next section. The following command will create an socket-based executable:

```
acecc HelloWorld.aCe -xskt -o HWorld
```

The other method of designating a multi-process executable is through the name of the output executable. If the output executable name is appended with the executable type, aCe will generate that type of executable. The following command will generate the same type of executable as the previous example:

```
acecc HelloWorld.aCe -o HWorld.skt
```

See the ‘acecc’ man page for more information on compiling aCe programs.

## Execution

A single-process aCe executable can be run like any standard Linux executable. The command is simply the name of the executable (e.g., a.out, HWorld, ...). A multi-process executable requires the command ‘acerun’. The command ‘acerun’ uses the file ‘/etc/nodelist’ to determine where to run a multi-process executable. It can also be used to run an executable on a subset of the nodes listed in the node list. In the next command below, ‘acerun’ will start processes on 4 nodes starting with the third node in the node list. (The first node in the node list is node 0.) It will put two processes per node until it runs out of nodes and then start again with node 2 repeatedly until 32 processes have been started.

```
acerun -n4 -p2 -s2 -t32 HWorld.skt
```

Like the ‘acecc’ command, ‘acerun’ has the ‘-x’ option, which allows an executable to be designated a particular type if it doesn’t have a type suffix. The following command does the same as the previous one if the executable has no prefix:

```
acerun -n4 -p2 -s2 -t32 -xskt HWorld
```

If there were arguments for the executable ‘HWorld’, they must follow the name on the command line:

```
acerun -n4 -p2 -s2 -t32 -xskt HWorld -t -x 365
```

All options for ‘mnr’ must go between the word ‘acerun’ and the name of the executable (in this case ‘HWorld’).

See the ‘acerun’ man page for more information on the multi-process run command.