CONDOR INSTALLATION GUIDE

Michael Litzkow

Computer Sciences Department
University of Wisconsin - Madison
mike@cs.wisc.edu

1. GENERAL

This document explains how to create and install *condor* from the source. **To do this, you will need the capability of becoming super user on all of the machines where you want condor to run.** We also assume that you are familiar with local procedures for creating UNIX® accounts, and allocating user and group id numbers. In addition, if you wish to use NFS® to share some of the *condor* executables and libraries between machines, you will need to be familiar with exporting and importing NFS file systems.

2. REQUIRED TOOLS

Building this version of *condor* requires use of *imake* and *makedepend* which are part of the X11 distribution. *Imake* is a program for creating ‘‘Makefiles’’ which are customized for a particular platform and local environment. This is accomplished by combining information in a generic description of a ‘‘Makefile’’, called an ‘‘Imakefile’’, along with configuration files which have been customized for your particular platform and your local site. *Makedepend* is a program for adding dependencies to ‘‘Makefiles’’ for C programs which use various include files. These tools are available from MIT as part of the X11 distribution. In case you do not have them, they are included in this distribution in the "CONDOR/imake_tools" directory. Instructions for building them are given in a separate document, ‘‘Building Imake, Cpp, and Makedepend’’.

3. OPERATIONS IN A HETEROGENEOUS ENVIRONMENT

Most local area networks contain a variety of machine and operating system types. We refer to a particular combination of machine architecture and UNIX variant as a ‘‘platform’’. To maximize the sharing of resources, *condor* attempts to provide some interoperability between the various platforms. This does add some complication to the *condor* code, and will require some consideration by the person creating and installing the *condor* executables.

Condor is designed so that all of the various machines may reside in a single ‘‘resource’’ pool. All of the *condor* daemons communicate through XDR® library routines, and are thus compatible even between machines which use different byte ordering. Users of a machine of any particular architecture and operating system type will be able to submit jobs to be run on other architecture and operating system combinations. (Of course those jobs will need to be compiled and linked for the appropriate target architecture/operating system combination.)[1]

®UNIX is a trademark of AT&T.

®NFS is a trademark of Sun Microsystems.

®XDR is a trademark of Sun Microsystems.

[1]Currently it is not possible to submit jobs compiled for R6000 machines from other platforms, nor is it possible to submit jobs compiled for other platforms from R6000 machines.

Supported Architecture and Operating System Combinations

Architecture	Operating System	Description
R6000®	AIX31®	IBM R6000 running AIX 3.1†
MIPS®	ULTRIX®	DECstation 3100 running Ultrix 3.0+
MIPS	ULTRIX40	DECstation 3100 running Ultrix 4.0+
SPARC®	SUNOS41®	Sun 3 running SunOs4.1
MC68020®	SUNOS41	Sun 4 running SunOs4.1
I386®	DYNIX®	Sequent symmetry running Dynix
VAX®	ULTRIX	Vax running Ultrix 3.0+
MC68020	BSD43®	HP 9000 running BSD4.3+
SGI	IRIX332®	Silicon Graphics 4D running IRIX 3.3.1+

4. PLANNING

4.1. Sharing Source

If you use NFS, you can maintain the *condor* source on a single machine. It is then possible to build object trees for the various platforms either on the source machine or on the various machines for which those objects are being built. If you do not use NFS, you will need to distribute the source tree to the appropriate machines using ‘‘rcp’’ or ‘‘rdist’’.

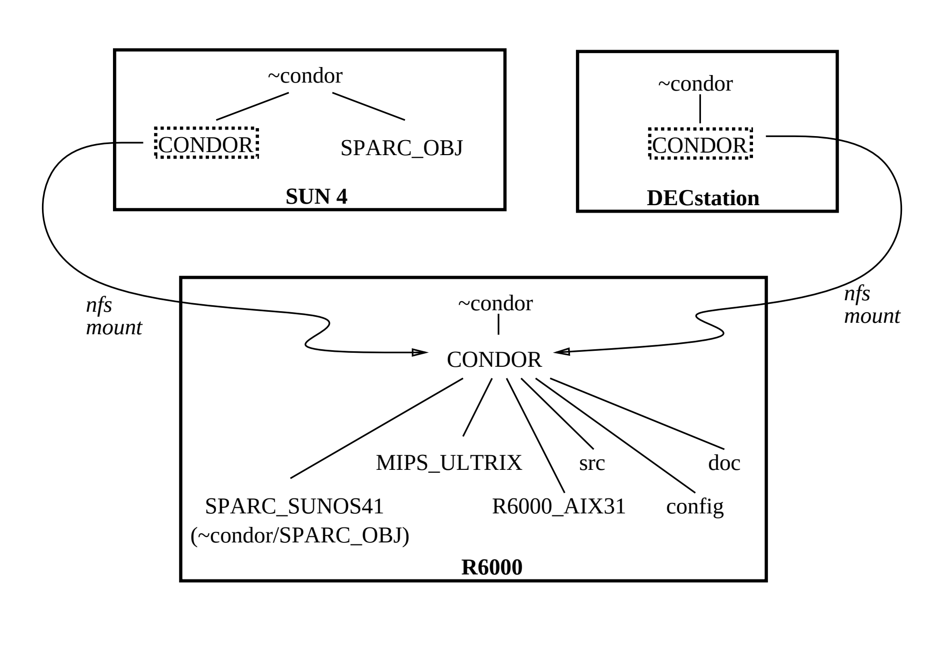

Figure 1

† Checkpointing is not currently supported on R6000/AIX machines.

®R6000 and AIX are trademarks of International Business Machines Corporation.

®VAX and ULTRIX are trademarks of Digital Equipment Corporation.

®MIPS is a trademark of Mips Computer Systems.

®SPARC is a trademark of Sparc International.

®SunOS is a trademark of Sun Microsystems.

®HP9000 is a trademark of Hewlett Packard.

®MC68020 is a trademark of Motorola Semiconductor Corporation.

®Symmetry and Dynix are a trademarks of Sequent Computer Systems.

®BSD 4.3 is a trademark of the Regents of the University of California.

®IRIX is a trademark of Silicon Graphics Incorporated.

Figure 1 illustrates sharing of the condor source, configuration, and document directories between three different platforms. In this case the MIPS_ULTRIX and R6000_AIX31 object directories and all the non-object directories reside physically on the R6000. The SPARC_SUNOS41 object directory resides physically on the SUN 4. Both the SUN 4 and the DECstation mount ‘‘~condor/CONDOR’’ from the R6000. Note that ‘‘SPARC_SUNOS41’’ is a symbolic link to ‘‘~condor/SPARC_OBJ’’. From the point of view of the R6000, this is a dangling symbolic link, but from the point of view of the SUN 4 it is valid and points into the local disk.

4.2. Sharing Objects

Once you have compiled executables suitable for the machines you wish to include in your *condor* pool, you will need to make those executables available on the member machines. Following these instructions for building ‘‘condor’’ for a particular platform type will result in the creation of a platform specific object directory under the ‘‘CONDOR’’ directory. Examples of such platform specific directories are the ‘‘MIPS_ULTRIX40’’ and ‘‘R6000_AIX31’’ shown in figure 2. Each platform specific object directory will contain several subdirectories for building the various *condor* programs and libraries. Also each platform specific object directory will contain a special subdirectory called ‘‘release_dir’’. Copies of all the completely linked programs and archived libraries (but no ‘‘.o’’ files) will be placed there. In the following discussion the pathname of this platform specific ‘‘release_dir’’ will be referred to as <RELEASE_DIR>. Another very important pathname is the one which you will set up for your users to access the *condor* software. In the example shown in figure 2 this is ‘‘/usr/uw/condor’’. For the following discussion, this ‘‘well known’’ directory will be referred to as <INSTALL_DIR>. Note that <RELEASE_DIR> will be a different pathname for every platform type, while <INSTALL_DIR> should be same pathname on every platform. When you install *condor* you will need to make the software in <RELEASE_DIR> available to your users under the name <INSTALL_DIR>. There are a number of possible ways to do this. To include the machine where you did the compilation in the pool, you might want to make <INSTALL_DIR> a symbolic link to <RELEASE_DIR>. For other machines in the pool, you might choose to use the compilation machine as a fileserver; in this case you can use NFS to mount the appropriate <RELEASE_DIR> as <INSTALL_DIR> on those machines. If the machines to be in the *condor* pool already have some of their executables remotely mounted from fileservers, you might want to copy <RELEASE_DIR> onto the fileservers using ‘‘rcp’’ or ‘‘rdist’’. In any case it is strongly suggested that you make the *condor* libraries and executables available to your users via the same path name on all machines.

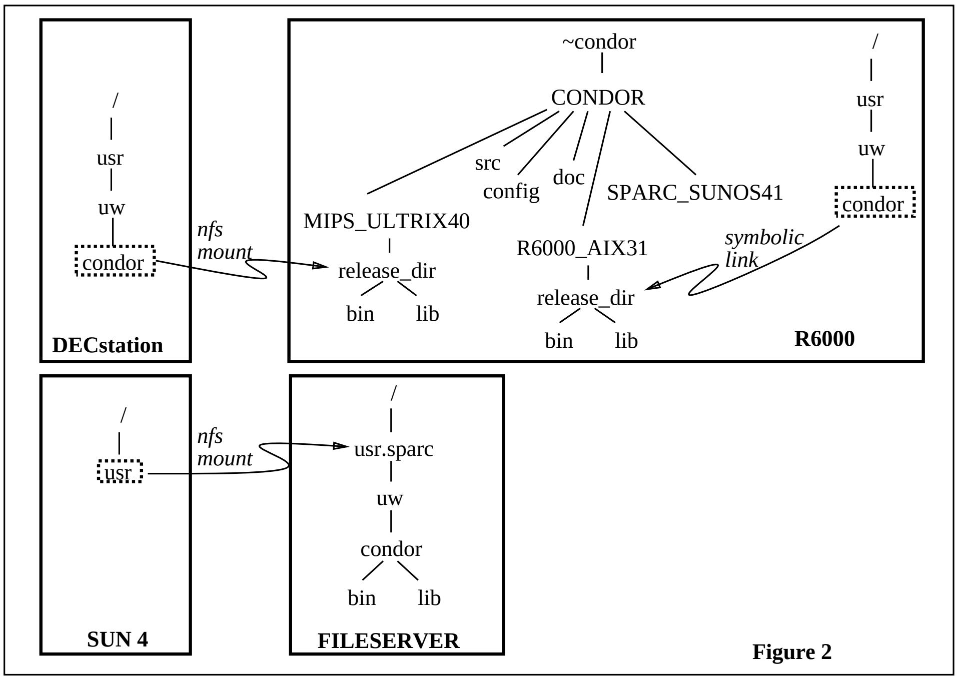

Figure 2

Figure 2 illustrates the user's access to the *condor* software on three different platforms. On the R6000, ''/usr/uw/condor'' is a symbolic link to ''~condor/CONDOR/R6000_AIX31/release_dir''. On the DECstation ''/usr/uw/condor'' is remotely mounted from the R6000 which is acting as a fileserver. On the DECstation ''/usr/uw/condor'' is remotely mounted from a dedicated fileserver.

4.3. Administrative Details

Each machine which is a member of the *condor* pool will need to have a ''condor'' account. If you use NFS, it will be necessary for all of the ''condor'' accounts to share common user and group id's. *Condor* must have its own group id. Group id's such as ''daemon'' or ''bin'' are not suitable for use by *condor*. Each ''condor'' account will need a home directory containing 3 subdirectories, ''log'', ''spool'', and ''execute''. A script is provided which will create these directories with the proper ownership and mode. If you choose to have these directories remotely mounted, be sure each *condor* machine has its own private version of these directories. Each ''condor'' account will have two files in the home directory called ''condor_config'' and ''condor_config.local''. It is intended that ''Condor_config'' be shared via NFS, but ''condor_config.local'' should be private. In this way you can use ''condor_config'' to globally configure your condor pool, and use ''condor_config.local'' to make changes for individual machines. Finally, each member machine will need to start the *condor* daemons at boot time. This will be done by placing an entry in ''/etc/rc'' or ''/etc/rc.local'' which starts the ''condor_master''. The master will determine which daemons should be run on its machine, and will monitor their health, restarting them and mailing system personnel about problems as necessary.

In addition to the member machines, you will need to designate one machine to act as the ''central manager''. The central manager will run two extra daemons which communicate with and coordinate the daemons on all member machines. These daemons will also be started and monitored by the master.

5. CREATING AND DISTRIBUTING EXECUTABLES

(1) Create a user account for ''condor'', on the machine where you want to maintain the condor source files, and set up a condor group there as well. Change directory to the condor home directory and run as ''condor'', e.g. ''su condor''.

(2) Extract the ''~condor/CONDOR'' directory from the distribution file, e.g. ''uncompress Condor.tar.Z'', then ''tar xf Condor.tar''.

(3) If you wish to make executables for a platform type other than the machine where you have extracted the tape, you will need to either copy the files to the ''compilation'' machine, or preferably remotely mount them via NFS. In any case, all the condor files should have owner ''condor'', and group ''condor''. You should always be running as ''condor'' when you make condor executables.

(4) Determine which versions of *imake*, *cpp*, and *makedepend* you will use. If you want to use the versions supplied with this distribution, you should build and install them now. Regardless where you obtain your *imake*, you will want to invoke it with a standard set of parameters and with a particular environment so that it behaves as assumed in these instructions. We recommend that you set up an alias for the appropriate invocation on your compilation machine. Ideally this would be done in the *condor* login shell script. Some versions of *imake* automatically invoke *make* on the generated ''Makefile'', while others do not. These instructions assume that *make* will be invoked separately, and your alias can include the ''-s Makefile'' to achieve this behavior if it is is not the default for your version of *imake*. Also, you will need to tell *imake* where your *condor* configuration files are by using the ''-I'' flag. For example on a machine where the default *imake* behavior is to execute *make*, your alias might look like

```
alias imake 'imake -s Makefile -I/u/condor/CONDOR/config'
```

Since *imake* expects very specific behavior from the *cpp* program it invokes, it may be necessary to tell *imake* to use a special version. You can do this by placing the pathname in your environment variable ''IMAKECPP''. Again, this should be done in the *condor* login script, and the command might look like setenv IMAKECPP /usr/lpp/X11/Xamples/util/cpp/cpp

(5) ‘‘Cd’’ to the ‘‘CONDOR/config’’ directory, and edit the ‘‘site.def’’ file.

Top
: Pathname of directory where you will keep condor sources and objects. Use of the ‘‘$(TILDE)’’ macro is encouraged here so that you can build on several platforms without requiring ˜condor to be the same pathname on all of them.

    $(TILDE)/CONDOR

  should be appropriate in most cases.

InstallDir
: Pathname where you will ultimately install condor executables and libraries. This is the pathname referred to as <INSTALL_DIR> in these instructions.

TmpDir
: Pathname of directory where you wish temporary files to be placed during the condor building and testing process.

CFlags
: Global flags to the C compiler which you will want used on all platforms, e.g. ‘‘-g’’.

CentralMgrHost
: Hostname of machine where you will run condor’s central manager processes.

CondorAdministrator
: Mailing address of person who will be responsible for managing and maintaining condor at your site. Condor will send mail about problems to this address.

YpLib
: If you run ‘‘Yellow Pages’’, set this to the pathname of the associated library, otherwise leave it blank.

(6) ‘‘Cd’’ to the ‘‘CONDOR/config’’ directory, and edit the appropriate platform configuration file, e.g. if you want to build condor for a Sun 4, edit ‘‘SPARC_SUNOS41.cf’’. Only a few items will need to be changed.

Tilde
: Pathname of ˜condor on the compilation machine. This may be different from ˜condor on the machine where you store the condor source directories, see figure 1.

SimpleLibC
: Pathname of the C library.

XLibPath
: Pathname of the X library. If you don’t run the X window system on machines of this platform type, don’t worry about the X library pathname, but set ‘‘HAS_X’’ to ‘‘NO’’.

XExtLibPath
: Pathname of the X extension library. If you don’t have an X extension library on machines of this platform type, don’t worry about this pathname, but set ‘‘HAS_X_EXTENSIONS’’ to ‘‘NO’’.

MkDepend
: Pathname of the *makedepend* program you intend to use. This could be a previously existing X11 version, or the shell script provided in this distribution tape.

TypicalMemory
: The amount of physical memory which will be typical on platforms of this type at your installation. Information about individual machines which don’t fit the norm can be customized later.

(7) If you are setting up Condor on an R6000 system, you should set up alternate entry points for both the C and FORTRAN compilers to aid in the correct linking of condor programs. Note: building the condor test software depends on these entry points.

C compiler

Edit the ‘‘/etc/xlc.cfg’’ file and create two new configuration stanzas, one called ‘‘condorcc’’ and another called ‘‘ckptcc’’. Both stanzas should be copied from ‘‘bsdcc’’ with the following changes. For ‘‘condorcc’’ change

/lib/libc.a

to

<INSTALL_DIR>/lib/libcondor.a

For ‘‘ckptcc’’ change

/lib/libc.a

to

<INSTALL_DIR>/lib/libckpt.a

For both ‘‘ckptcc’’ and ‘‘condorcc’’ change

/lib/crt0.o

to

<INSTALL_DIR>/lib/condor_rt0.o

and add

-bI:<INSTALL_DIR>/lib/syscall.exp

to the ‘‘libraries’’ line. Finally, create links from ‘‘/bin/bsdcc’’ to ‘‘/bin/condorcc’’ and ‘‘/bin/ckptcc’’.

FORTRAN compiler

Edit the ‘‘/etc/xlf.cfg’’ file and create two new configuration stanzas, one called ‘‘condorxlf’’ and another called ‘‘ckptxlf’’. Both stanzas should be copied from ‘‘xlf’’ with the following changes. For ‘‘condorxlf’’ change

/lib/libc.a

to

<INSTALL_DIR>/lib/libcondor.a

For ‘‘ckptxlf’’ change

/lib/libc.a

to

<INSTALL_DIR>/lib/libckpt.a

For both ‘‘ckptxlf’’ and ‘‘condorxlf’’ change

/lib/crt0.o

to

<INSTALL_DIR>/lib/condor_rt0.o

and add

-bI:<INSTALL_DIR>/lib/syscall.exp

to the ‘‘libraries’’ line. Finally, create links from ‘‘usr/bin/xlf’’ to ‘‘usr/bin/condorxlf’’ and ‘‘usr/bin/ckptxlf’’.

The ‘‘condorcc’’ and ‘‘condorxlf’’ entries will link programs for execution by condor. The ‘‘ckptcc’’ and ‘‘ckptxlf’’ entries will link programs for local execution with checkpointing. Example ‘‘xlc.cfg’’ and ‘‘xlf.cfg’’ files are included in the ‘‘config’’ directory.

(8) Create an object tree for the specific platform type for which you are building executables. If you have the condor source remotely mounted, and want to use the trick shown in figure 1 to build your executables on the physical disk of the compilation machine, set up the directory on the compilation machine and the symbolic link now. For the example shown, the commands executed on the Sun 4 were

```
mkdir ˜condor/SPARC_OBJ
ln -s ˜condor/SPARC_OBJ ˜condor/CONDOR/SPARC_SUNOS41
```

To build the object tree, ‘‘cd’’ to the ‘‘˜condor/CONDOR’’ directory and first run ‘‘imake’’ then ‘‘make’’.

(9) If you are building executables for an R6000 system, you will need to make <RELEASE_DIR> accessible as <INSTALL_DIR> on the compilation machine at this time. This is because some of the *condor* programs will be built using a shared library. The AIX load routine will need to be able to find that library at execution time and will search for it at a pathname compiled into the affected program. It is therefore desirable that this pathname is relative to <INSTALL_DIR> rather than <RELEASE_DIR> so that if you later copy executables to a machine other than the compilation machine, that library can still be found. The ‘‘Imakefiles’’ supplied with this distribution are set up to do it that way.

(10) ‘‘Cd’’ to the newly created object directory for the platform type of interest, and compile all of condor by running ‘‘make release’’.

(11) Make condor available to all of the machines you wish to have in your pool as appropriate for your site. This may mean creating a symbolic link, distributing to a fileserver, or granting permission to other machines to mount the condor software via NFS. See figure 2.

(12) You will also need to install the *condor* man pages. These will be found in ‘‘CONDOR/doc/man/{manl,catl}’’. The exact commands will vary somewhat depending on the situation at your site. If you mount your man pages on a shared fileserver, they may look something like this:

```
rcp manl/* <fileserver>:/usr/man/manl
rcp catl/* <fileserver>:/usr/man/catl
```

(13) To make and install executables for other platforms, go back to step 3.

6. STARTING CONDOR AND ADDING MEMBERS TO THE POOL

Complete the following steps on each machine you want to add to the condor resource pool. Add the machine which will act as the ‘‘central manager’’ first. **N.B. all of the steps in setting up a member of the condor pool will require you to operate as the super user.**

(1) Create an account for ‘‘condor’’ on the member machine. Be sure to use the same user and group id’s on all member machines.

(2) If you are planning to access the condor executables on this machine via a remotely mounted file system, make sure that file system is currently mounted, and that there is an appropriate entry in ‘‘/etc/fstab’’ so that it will get mounted whenever the machine is booted.

For example on a Sun 4 running SunOs 4.1 the fstab entry might look something like:

```
<fileserver>:<some path>/SPARC_SUNOS41/release <RELEASEDIR> nfs ro
0 0
```

(3) Run the script ‘‘condor_init’’. This will link ‘‘condor_config’’ to a site specific version of that file, and create the ‘‘log’’, ‘‘spool’’, and ‘‘execute’’ directories with correct ownership and permissions.

(4) Run the script ‘‘condor_on’’. This will create and edit ‘‘condor_config.local’’ setting ‘‘START_DAEMONS’’ to ‘‘True’’ so that the condor daemons are able to run, then it will actually start them.

(5) At this point the member machine should be fully operational. On all machines you should find the ‘‘condor_master’’, ‘‘condor_startd’’, and ‘‘condor_schedd’’ running. Machines which run the X window system, should also be running the ‘‘condor_kbdd’’. Additionally the ‘‘central manager’’ machine should be running the ‘‘condor_collector’’ and ‘‘condor_negotiator’’. You can check to see that the proper daemons are running with

```
ps -ax | egrep condor
```

You should also run ‘‘condor_status’’ to see that the new machine shows up in the resource pool. If you wish to run some trivial jobs to check operation of all the *condor* software, example user programs and ‘‘job description’’ files have been compiled and are provided in the ‘‘condor_test_suite_C’’ and ‘‘condor_test_suite_F’’ directories.

(6) Add lines to ‘‘/etc/rc’’ or ‘‘/etc/rc.local’’ which will start ‘‘condor_master’’ at boot time.

The entry will look something like this:

```
if [ -f /usr/uw/condor/bin/condor_master ]; then
    /usr/uw/condor/bin/condor_master; echo -n ' condor'   >/dev/console
fi
```

Note: do not attempt to run this command now, condor is already running.

7. Copyright Information

Authors: Allan Bricker, Michael J. Litzkow, and others.
University of Wisconsin, Computer Sciences Dept.