Fortran Interface

```
include 'fpvm3.h'
```

Process Control

```
call pvmfmytid( tid )
call pvmfexit( info )
call pvmfkill(  tid, info )
call pvmfspawn( task, flag, where,
                ntask, tids, numt )
```

Spawn flag options		MEANING
PVMDEFAULT	0	don't care where
PVMHOST	1	"where" contains host
PVMARCH	2	"where" contains arch
PVMDEBUG	4	start tasks with debug on

Information

```
call pvmfparent( tid )
call pvmfpstat(  tid, pstat )
call pvmfmstat(  host, mstat )
call pvmfconfig( nhost, narch, info)
call pvmftasks(  which, ntask, info)
```

Dynamic Configuration

```
call pvmfaddhost( host, info )
call pvmfdelhost( host, info )
```

Dynamic Process Groups

```
call pvmfjoingroup( group, inum )
call pvmflvgroup(   group, info )
call pvmfgsize(     group, size, info )
call pvmfgettid(    group, inum, tid )
call pvmfgetinst(   group, tid, inum )
call pvmfbarrier(   group, count, info )
call pvmfbcast(     group, msgtag, info )
```

Message Buffers

```
call pvmfmkbuf( encoding, bufid )
call pvmffreebuf( bufid, info )
call pvmfgetsbuf( bufid )
call pvmfgetrbuf( bufid )
call pvmfsetsbuf( bufid, oldbuf )
call pvmfsetrbuf( bufid, oldbuf )
call pvmfinitsend( encoding, bufid )
```

Encoding options		MEANING
PVMDEFAULT	0	XDR if needed
PVMRAW	1	no encoding
PVMINPLACE	2	data left in place

Signalling

```
call pvmfsendsig( tid, signum, info )
call pvmfnotify(about, msgtag, ntask, tids, info)
```

Sending

```
call pvmfpack( what, xp, nitem, stride, info )
```

Pack/unpack what options			
STRING	0	REAL4	4
BYTE1	1	COMPLEX8	5
INTEGER2	2	REAL8	6
INTEGER4	3	COMPLEX16	7

```
call pvmfsend(  tid,   msgtag, info )
call pvmfmcast( ntask, tids, msgtag, info )
```

Receiving

```
call pvmfrecv(  tid, msgtag, bufid )
call pvmfprobe( tid, msgtag, bufid )
call pvmfnrecv( tid, msgtag, bufid )
call pvmfbufinfo(bufid, bytes, msgtag, tid, info)

call pvmfunpack( what, xp, nitem, stride, info )
```

Error Handling

```
call pvmfperror(msg, info)
call pvmfserror(set, oldset)
```

Declarations

INTEGER	about, count, encoding, flag, info
INTEGER	inum, bufid, msgtag, mstat
INTEGER	narch, nhost, nitem, ntask, numt
INTEGER	oldbuf, oldset, pstat, set, stride
INTEGER	signum, tid, tids(ntask),what, which
CHARACTER	group, host, msg, task, where
'WHAT'	xp(nitem*stride)

ERROR CODE		MEANING
PvmOk	0	okay
PvmBadParam	-2	bad parameter
PvmMismatch	-3	barrier count mismatch
PvmNoData	-5	read past end of buffer
PvmNoHost	-6	no such host
PvmNoFile	-7	no such executable
PvmNoMem	-10	can't get memory
PvmBadMsg	-12	can't decode received msg
PvmSysErr	-14	pvmd not responding
PvmNoBuf	-15	no current buffer
PvmNoSuchBuf	-16	bad message id
PvmNullGroup	-17	null group name is illegal
PvmDupGroup	-18	already in group
PvmNoGroup	-19	no group with that name
PvmNotInGroup	-20	not in group
PvmNoInst	-21	no such instance in group
PvmHostFail	-22	host failed
PvmNoParent	-23	no parent task
PvmNotImpl	-24	function not implemented
PvmDSysErr	-25	pvmd system error
PvmBadVersion	-26	pvmd-pvmd protocol mismatch
PvmOutOfRes	-27	out of resources
PvmDupHost	-28	host already configured
PvmCantStart	-29	failed to exec new slave pvmd
PvmAlready	-30	already doing operation
PvmNoTask	-31	no such task
PvmNoEntry	-32	no such (group,instance)
PvmDupEntry	-33	(group,instance) already exists

Parallel Virtual Machine

Quick Reference Guide

Release 3.1

May 21, 1993

University of Tennessee

Oak Ridge National Laboratory

Emory University

C Interface

```
#include "pvm3.h"
```

Process Control

```
int tid  = pvm_mytid(void)
int info = pvm_exit( void)
int info = pvm_kill( int tid)
int numt = pvm_spawn(char *task, char **argv,
                     int flag, char *where,
                     int ntask, int *tids)
```

Spawn flag options		MEANING
`PvmTaskDefault`	0	don't care where
`PvmTaskHost`	1	"where" contains host
`PvmTaskArch`	2	"where" contains arch
`PvmTaskDebug`	4	start tasks with debug on

Information

```
int tid   = pvm_parent(void)
int pstat = pvm_pstat( int tid)
int mstat = pvm_mstat( char *host)
int info  = pvm_config(int *nhost, int *narch,
                       struct hostinfo **hostp)
int info  = pvm_tasks(int which, int *ntask,
                      struct taskinfo **taskp)
```

Dynamic Configuration

```
int info = pvm_addhosts(char **hosts, int nhost,
                        int *infos)
int info = pvm_delhosts(char **hosts, int nhost)
                        int *infos)
```

Dynamic Process Groups

```
int inum = pvm_joingroup(char *group)
int info = pvm_lvgroup(  char *group)
int size = pvm_gsize(    char *group)
int tid  = pvm_gettid(   char *group, int inum)
int inum = pvm_getinst(  char *group, int tid)
int info = pvm_barrier(  char *group, int count)
int info = pvm_bcast(    char *group, int msgtag)
```

Signalling

```
int info = pvm_sendsig(int tid, int signum)
int info = pvm_notify(int about, int msgtag,
                      int ntask, int *tids)
```

About options		MEANING
`PvmTaskExit`	1	notify if task exit
`PvmHostDelete`	2	notify if deletion
`PvmHostAdd`	3	notify if addition

Message Buffers

```
int bufid  = pvm_mkbuf(int encoding)
int info   = pvm_freebuf(int bufid)
int bufid  = pvm_getsbuf(void)
int bufid  = pvm_getrbuf(void)
int oldbuf = pvm_setsbuf(int bufid)
int oldbuf = pvm_setrbuf(int bufid)
int bufid  = pvm_initsend(int encoding)
```

Encoding options		MEANING
`PvmDataDefault`	0	XDR if needed
`PvmDataRaw`	1	no encoding
`PvmDataInPlace`	2	data left in place

Sending

```
int info = pvm_pkbyte(  char   *cp, int cnt, int std )
int info = pvm_pkcplx(  float  *xp, int cnt, int std )
int info = pvm_pkdcplx( double *zp, int cnt, int std )
int info = pvm_pkdouble(double *dp, int cnt, int std )
int info = pvm_pkfloat( float  *fp, int cnt, int std )
int info = pvm_pkint(   int    *np, int cnt, int std )
int info = pvm_pklong(  long   *np, int cnt, int std )
int info = pvm_pkshort( short  *np, int cnt, int std )
int info = pvm_pkstr(   char   *cp )

int info = pvm_send(  int  tid, int msgtag )
int info = pvm_mcast( int *tids, int ntask, int msgtag )
```

Receiving

```
int bufid = pvm_recv(   int tid, int msgtag )
int bufid = pvm_probe(  int tid, int msgtag )
int bufid = pvm_nrecv(  int tid, int msgtag )
int info = pvm_bufinfo( int bufid, int *bytes,
                        int *msgtag, int *tid)

int info = pvm_upkbyte(  char   *cp, int cnt, int std )
int info = pvm_upkcplx(  float  *xp, int cnt, int std )
int info = pvm_upkdcplx( double *zp, int cnt, int std )
int info = pvm_upkdouble(double *dp, int cnt, int std )
int info = pvm_upkfloat( float  *fp, int cnt, int std )
int info = pvm_upkint(   int    *np, int cnt, int std )
int info = pvm_upklong(  long   *np, int cnt, int std )
int info = pvm_upkshort( short  *np, int cnt, int std )
int info = pvm_upkstr(   char   *cp )
```

Error Handling

```
int info   = pvm_perror(char *msg)
int oldhow = pvm_serror(int how)
```

Error Logs

stderr and stdout from all spawned tasks
is redirected to `/tmp/pvml.<uid>`
on the host where PVM was started.

Obtaining PVM

To receive a copy of PVM send email to `netlib@ornl.gov`
with the message `send index from pvm3` .
Questions can be sent to `pvm@msr.epm.ornl.gov` .

Starting PVM

Syntax

`pvmd3 hostfile`
`pvm` (starts console)

Host File Syntax

`hostname [ options ]`
...
Hostname '*' used to set defaults.
Comment lines begin with '#'.

Options:

OPTION	MEANING	DEFAULT
`lo=`	Unix Loginname	Name on Master
`pw`	Ask for Password	Don't ask
`ep=`	Component Path	$HOME/pvm3/bin/%

Multiple path elements are separated by colons (":").
'%' in path expands to architecture string.

Options for hosts to be added later are specified
by preceding hostname with '&'.

PVM Console Commands

`help [command]`	– get information about commands
`conf`	– lists hosts in virtual machine
`add host`	– add host to virtual machine
`delete host`	– delete host from virtual machine
`spawn file [args]`	– spawn process
`ps [ax]`	– lists processes on virtual machine
`pstat tid`	– show status of process
`mstat host`	– show status of host
`version`	– print libpvm version
`id`	– print console tid
`sig num tid`	– send signal num to process
`kill tid`	– terminate a process
`reset`	– kill all processes and reset PVM
`halt`	– kill all pvmds and console
`quit`	– exit console (PVM continues)

Compiling PVM Applications

```
cc -o task myprog.c libpvm3.a
f77 -o task myprog.f libfpvm3.a libpvm3.a
```

For groups add `libgpvm3.a` before `libpvm3.a`