Parallel Programming Laboratory
University of Illinois at Urbana-Champaign

Converse Programming Tutorial

Converse Parallel Programming Environment was developed as a group effort at Parallel Programming Laboratory, University of Illinois at Urbana-Champaign.

Version 1.0

University of Illinois
Charm++/Converse Parallel Programming System Software
Non-Exclusive, Non-Commercial Use License

Upon execution of this Agreement by the party identified below ("Licensee"), The Board of Trustees of the University of Illinois ("Illinois"), on behalf of The Parallel Programming Laboratory ("PPL") in the Department of Computer Science, will provide the Charm++/Converse Parallel Programming System software ("Charm++") in Binary Code and/or Source Code form ("Software") to Licensee, subject to the following terms and conditions. For purposes of this Agreement, Binary Code is the compiled code, which is ready to run on Licensee's computer. Source code consists of a set of files which contain the actual program commands that are compiled to form the Binary Code.

1. The Software is intellectual property owned by Illinois, and all right, title and interest, including copyright, remain with Illinois. Illinois grants, and Licensee hereby accepts, a restricted, non-exclusive, non-transferable license to use the Software for academic, research and internal business purposes only, e.g. not for commercial use (see Clause 7 below), without a fee.

2. Licensee may, at its own expense, create and freely distribute complimentary works that interoperate with the Software, directing others to the PPL server (`http://charm.cs.uiuc.edu`) to license and obtain the Software itself. Licensee may, at its own expense, modify the Software to make derivative works. Except as explicitly provided below, this License shall apply to any derivative work as it does to the original Software distributed by Illinois. Any derivative work should be clearly marked and renamed to notify users that it is a modified version and not the original Software distributed by Illinois. Licensee agrees to reproduce the copyright notice and other proprietary markings on any derivative work and to include in the documentation of such work the acknowledgement:

   > "This software includes code developed by the Parallel Programming Laboratory in the Department of Computer Science at the University of Illinois at Urbana-Champaign."

   Licensee may redistribute without restriction works with up to 1/2 of their non-comment source code derived from at most 1/10 of the non-comment source code developed by Illinois and contained in the Software, provided that the above directions for notice and acknowledgement are observed. Any other distribution of the Software or any derivative work requires a separate license with Illinois. Licensee may contact Illinois (`kale@cs.uiuc.edu`) to negotiate an appropriate license for such distribution.

3. Except as expressly set forth in this Agreement, THIS SOFTWARE IS PROVIDED "AS IS" AND ILLINOIS MAKES NO REPRESENTATIONS AND EXTENDS NO WARRANTIES OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO WARRANTIES OR MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, OR THAT THE USE OF THE SOFTWARE WILL NOT INFRINGE ANY PATENT, TRADEMARK, OR OTHER RIGHTS. LICENSEE ASSUMES THE ENTIRE RISK AS TO THE RESULTS AND PERFORMANCE OF THE SOFTWARE AND/OR ASSOCIATED MATERIALS. LICENSEE AGREES THAT UNIVERSITY SHALL NOT BE HELD LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, OR INCIDENTAL DAMAGES WITH RESPECT TO ANY CLAIM BY LICENSEE OR ANY THIRD PARTY ON ACCOUNT OF OR ARISING FROM THIS AGREEMENT OR USE OF THE SOFTWARE AND/OR ASSOCIATED MATERIALS.

4. Licensee understands the Software is proprietary to Illinois. Licensee agrees to take all reasonable steps to insure that the Software is protected and secured from unauthorized disclosure, use, or release and will treat it with at least the same level of care as Licensee would use to protect and secure its own proprietary computer programs and/or information, but using no less than a reasonable standard of care. Licensee agrees to provide the Software only to any other person or entity who has registered with Illinois. If licensee is not registering as an individual but as an institution or corporation each member of the institution or corporation who has access to or uses Software must agree to and abide by the terms of this license. If Licensee becomes aware of any unauthorized licensing, copying or use of the Software, Licensee shall promptly notify Illinois in writing. Licensee expressly agrees to use the Software only in the manner and for the specific uses authorized in this Agreement.

5. By using or copying this Software, Licensee agrees to abide by the copyright law and all other applicable laws of the U.S. including, but not limited to, export control laws and the terms of this license. Illinois shall have the right to terminate this license immediately by written notice upon Licensee's breach of, or non-compliance with, any terms of the license. Licensee may be held legally responsible for any copyright infringement that is caused or encouraged by its failure to abide by the terms of this license. Upon termination, Licensee agrees to destroy all copies of the Software in its possession and to verify such destruction in writing.

6. The user agrees that any reports or published results obtained with the Software will acknowledge its use by the appropriate citation as follows:

   > "Charm++/Converse was developed by the Parallel Programming Laboratory in the Department of Computer Science at the University of Illinois at Urbana-Champaign."

   Any published work which utilizes Charm++ shall include the following reference:

   > "L. V. Kale and S. Krishnan. Charm++: Parallel Programming with Message-Driven Objects. In 'Parallel Programming using C++' (Eds. Gregory V. Wilson and Paul Lu), pp 175-213, MIT Press, 1996."

   Any published work which utilizes Converse shall include the following reference:

   > "L. V. Kale, Milind Bhandarkar, Narain Jagathesan, Sanjeev Krishnan and Joshua Yelon. Converse: An Interoperable Framework for Parallel Programming. Proceedings of the 10th International Parallel Processing Symposium, pp 212-217, April 1996."

   Electronic documents will include a direct link to the official Charm++ page at `http://charm.cs.uiuc.edu/`

7. Commercial use of the Software, or derivative works based thereon, REQUIRES A COMMERCIAL LICENSE. Should Licensee wish to make commercial use of the Software, Licensee will contact Illinois (kale@cs.uiuc.edu) to negotiate an appropriate license for such use. Commercial use includes:

   (a) integration of all or part of the Software into a product for sale, lease or license by or on behalf of Licensee to third parties, or

   (b) distribution of the Software to third parties that need it to commercialize product sold or licensed by or on behalf of Licensee.

8. Government Rights. Because substantial governmental funds have been used in the development of Charm++/Converse, any possession, use or sublicense of the Software by or to the United States government shall be subject to such required restrictions.

9. Charm++/Converse is being distributed as a research and teaching tool and as such, PPL encourages contributions from users of the code that might, at Illinois' sole discretion, be used or incorporated to make the basic operating framework of the Software a more stable, flexible, and/or useful product. Licensees who contribute their code to become an internal portion of the Software agree that such code may be distributed by Illinois under the terms of this License and may be required to sign an "Agreement Regarding Contributory Code for Charm++/Converse Software" before Illinois can accept it (contact `kale@cs.uiuc.edu` for a copy).

UNDERSTOOD AND AGREED.
Contact Information:
The best contact path for licensing issues is by e-mail to `kale@cs.uiuc.edu` or send correspondence to:

Prof. L. V. Kale
Dept. of Computer Science
University of Illinois
201 N. Goodwin Ave
Urbana, Illinois 61801 USA
FAX: (217) 333-3501

Contents

Chapter 1

Introduction

1.1 CthThreads

The CthThread package, like most thread packages, provides basic functionality for creating threads, destroying threads, yielding, suspending, and awakening a suspended thread. In addition, it provides facilities whereby you can write your own thread schedulers.

Figure **??** demonstrates how to write a simple program that creates CthThreads. The `CthCreateMigratable` is used and it takes a handler, an argument pointer, and the stack size for the thread. This is demonstrated in the `initThreads` function on line number 43. Once the threads are created, they are pushed on the scheduler queue with the `CthAwaken` call, which only takes the `CthThread` as an argument. On being scheduled, the handler function is called.

In the example, each thread then calls `CthYield`, which directs control back to the scheduler and pushes the thread back onto the queue. Then in a loop, each thread calls `CthYieldPrio NUM_YIELD` times, with the queuing strategy and necessary parameters. The threads call this with priority 0 and 1, lower integers (but non-negative) indicating higher priority. The effect of yielding with priority is that the higher priority thread on the queue has precedence over the other threads and hence will be scheduled first, based on the greedy decision the scheduler makes.

After this loop completes, the `threadDone` is called by each thread, which increments a counter and quits the program when all threads are done.

```
1  #include <converse.h>
2  #include <stdlib.h>
3  CpvDeclare(int,msgSize);
4  CpvDeclare(int,exitHandler);
5  CpvDeclare(int,node0Handler);
6  CpvDeclare(int,node1Handler);
7  void startRing()
8  {
9    char *msg = (char *)CmiAlloc(CpvAccess(msgSize));
10   *((int *)(msg+CmiMsgHeaderSizeBytes)) = CpvAccess(msgSize);
11   CmiSetHandler(msg,CpvAccess(node1Handler));
12   CmiSyncSendAndFree(1, CpvAccess(msgSize), msg);
13 }
14 void ringFinished(char *msg)
15 {
16   CmiFree(msg);
17   //exit
18   void *sendmsg = CmiAlloc(CmiMsgHeaderSizeBytes);
19   CmiSetHandler(sendmsg,CpvAccess(exitHandler));
20   CmiSyncBroadcastAllAndFree(CmiMsgHeaderSizeBytes,sendmsg);
21 }
22 //We finished for all message sizes. Exit now
23 CmiHandler exitHandlerFunc(char *msg)
24 {
25     CmiFree(msg);
26     CsdExitScheduler();
27     return 0;
28 }
29 //Handler on Node 0
30 CmiHandler node0HandlerFunc(char *msg)
31 {
32         ringFinished(msg);
33         return 0;
34 }
35 CmiHandler node1HandlerFunc(char *msg)
36 {
37     CpvAccess(msgSize) = *((int *)(msg+CmiMsgHeaderSizeBytes));
38     CmiSetHandler(msg,CpvAccess(node0Handler));
39     CmiSyncSendAndFree(0,CpvAccess(msgSize),msg);
40     return 0;
41 }
42 CmiStartFn mymain()
43 {
44     CpvInitialize(int,msgSize);
45     CpvAccess(msgSize)= 512 + CmiMsgHeaderSizeBytes;
46     CpvInitialize(int,exitHandler);
47     CpvAccess(exitHandler) = CmiRegisterHandler((CmiHandler) exitHandlerFunc);
48     CpvInitialize(int,node0Handler);
49     CpvAccess(node0Handler) = CmiRegisterHandler((CmiHandler) node0HandlerFunc);
50     CpvInitialize(int,node1Handler);
51     CpvAccess(node1Handler) = CmiRegisterHandler((CmiHandler) node1HandlerFunc);
52     if (CmiMyPe() == 0)
53         startRing();
54     return 0;
55 }
56 int main(int argc,char *argv[])
57 {
58     ConverseInit(argc,argv,(CmiStartFn)mymain,0,0);
59     return 0;
60 }
```

Figure 1.1: A Pingpong Example using Converse Handler

```
#include <stdio.h>
#include "converse.h"

#define HIGH_PRIO 0
#define LOW_PRIO 1
#define NUM_YIELD 10

int endCounter = 0;

//determine completion based on threads calling it
void threadDone() {
  endCounter++;
  if (endCounter == 2) CsdExitScheduler();
}

//worker function for worker1, yields with a low priority
void worker1Work(void* msg) {
  printf("start worker1\n");
  CthYield();
  printf("worker1 resumed first time\n");
  unsigned int prio  = LOW_PRIO;
  for(int i = 0; i < NUM_YIELD; i++) {
    CthYieldPrio(CQS_QUEUEING_IFIFO,0,&prio);
    printf("worker1 resumed %dth time\n",i);
  }
  threadDone();
}

//worker function for worker2, yields with a high priority
void worker2Work(void* msg) {
  printf("start worker2\n");
  CthYield();
  printf("worker2 resumed first time\n");
  unsigned int prio  = HIGH_PRIO;
  for(int i = 0; i < NUM_YIELD; i++) {
    CthYieldPrio(CQS_QUEUEING_IFIFO,0,&prio);
    printf("worker2 resumed %dth time\n",i);
  }
  threadDone();
}

//create two worker threads and push them on scheduler Q
void initThreads(int argc, char* argv[]) {
  printf("called initThreads\n");
  CthThread worker1 = CthCreateMigratable((CthVoidFn)worker1Work, 0, 160000);
  CthThread worker2 = CthCreateMigratable((CthVoidFn)worker2Work, 0, 160000);
  CthAwaken(worker1); CthAwaken(worker2);
}

int main(int argc, char* argv[]) {
  ConverseInit(argc, argv, initThreads, 0, 0);
}
```

Figure 1.2: CthThread Example