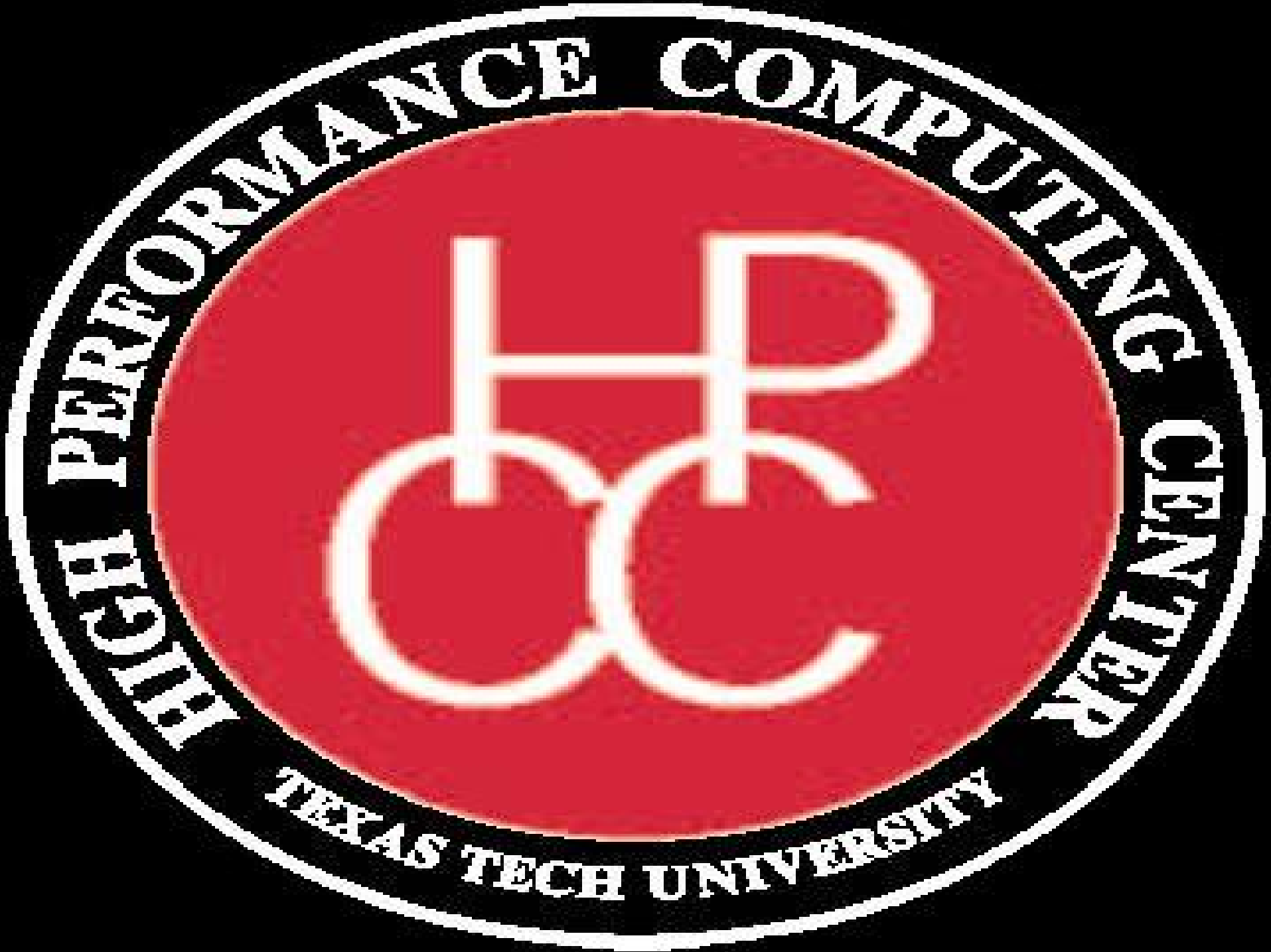

HIGH PERFORMANCE COMPUTING CENTER
TEXAS TECH UNIVERSITY

Greetings

```
/* greetings.c -- greetings program
 *
 * Send a message from all processes with rank != 0 to
   process 0.
 *    Process 0 prints the messages received.
 *
 * Input: none.
 * Output: contents of messages received by process 0.
 *
 * See Chapter 3, pp. 41 & ff in PPMPI.
 */
#include <stdio.h>
#include <string.h>
#include "mpi.h"
```

```
main(int argc, char* argv[]) {
   int       my_rank;       /* rank of process     */
   int       p;             /* number of processes  */
   int       source;         /* rank of sender      */
   int       dest;          /* rank of receiver    */
   int       tag = 0;        /* tag for messages    */
   char      message[100];  /* storage for message  */
   MPI_Status  status;       /* return status for   */
                              /* receive             */
               /* Start up MPI */
   MPI_Init(&argc, &argv);
               /* Find out process rank  */
   MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
               /* Find out number of processes */
   MPI_Comm_size(MPI_COMM_WORLD, &p);
   if (my_rank != 0) {
               /* Create message */
      sprintf(message, "Greetings from process %d!",
         my_rank);
      dest = 0;
```

```
            /* Use strlen+1 so that '\0' gets transmitted */
      MPI_Send(message, strlen(message)+1, MPI_CHAR,
         dest, tag, MPI_COMM_WORLD);
   } else {    /* my_rank == 0 */
      for (source = 1; source < p; source++) {
         MPI_Recv(message, 100, MPI_CHAR, source, tag,
         MPI_COMM_WORLD, &status);
         printf("%s\n", message);
      }
   }
            /* Shut down MPI */
   MPI_Finalize();
}           /* main */
```

Greetings Master

```
#include <stdio.h>
#include <string.h>
#include <mpi.h>

#define TRUE 1
#define FALSE 0
#define MASTER_RANK 0

main(argc, argv)
int argc;
char *argv[];
{
  int count, pool_size, my_rank, my_name_length, i_am_the_master =
FALSE;
  char my_name[BUFSIZ], master_name[BUFSIZ], send_buffer[BUFSIZ],
      recv_buffer[BUFSIZ];
  MPI_Status status;
```

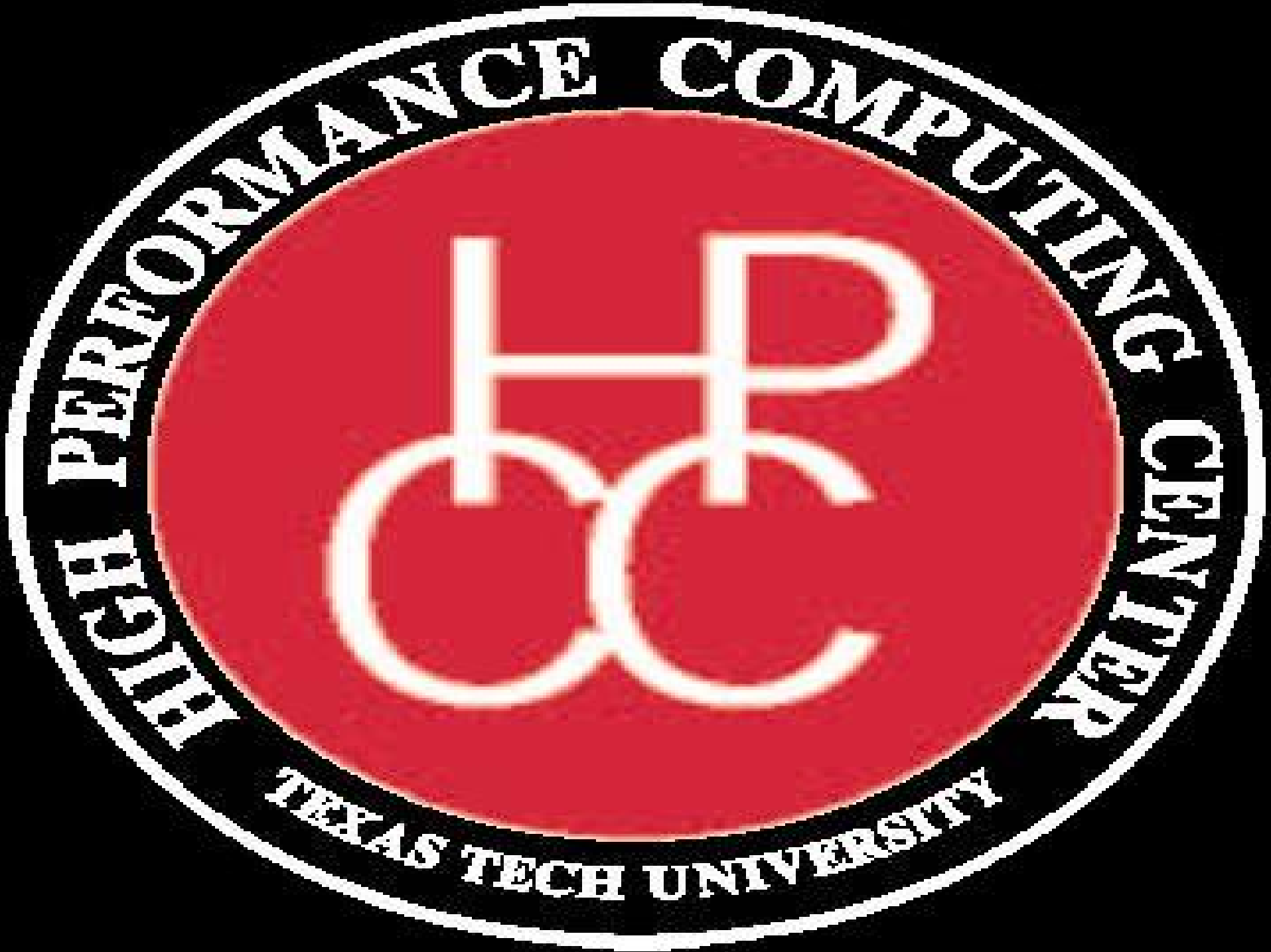
```
MPI_Init(&argc, &argv);
  MPI_Comm_size(MPI_COMM_WORLD, &pool_size);
  MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
  MPI_Get_processor_name(my_name, &my_name_length);

  if (my_rank == MASTER_RANK) {
    i_am_the_master = TRUE;
    strcpy (master_name, my_name);
  }

 MPI_Bcast(master_name, BUFSIZ, MPI_CHAR, MASTER_RANK, MPI_COMM_WORLD);

  sprintf(send_buffer, "hello %s, greetings from %s, rank = %d",
        master_name, my_name, my_rank);
  MPI_Send (send_buffer, strlen(send_buffer) + 1, MPI_CHAR,
          MASTER_RANK, 0, MPI_COMM_WORLD);

  if (i_am_the_master) {
    for (count = 1; count <= pool_size; count++) {
      MPI_Recv (recv_buffer, BUFSIZ, MPI_CHAR, MPI_ANY_SOURCE, MPI_ANY_TAG,
              MPI_COMM_WORLD, &status);
      printf ("%s\n", recv_buffer);
    }
  }
  MPI_Finalize();
}
```


HIGH PERFORMANCE COMPUTING CENTER
TEXAS TECH UNIVERSITY