

```
1: #include <stdio.h>
2: #include <stdlib.h>
3: #include <string.h>
4:
5: #include "adlb.h"
6:
7: #define CMDLINE 1
8:
9: int main(int argc, char *argv[])
10: {
11:     FILE *fp;
12:     int rc, i, done;
13:     char c, cmdbuffer[1024];
14:
15:     int am_server, am_debug_server;
16:     int num_servers, use_debug_server, aprintf_flag;
17:     MPI_Comm app_comm;
18:     int my_world_rank, my_app_rank;
19:
20:     int num_types = 1;
21:     int type_vect[2] = {CMDLINE};
22:
23:     int work_prio, work_type, work_handle[ADLB_HANDLE_SIZE], work_len,
24:         answer_rank;
25:     int req_types[4];
26:
27:     double start_time, end_time;
```



```
53:
54:     if (argc != 2) {
55:         printf("usage: %s <filename>\n", argv[0]);
56:         ADLB_Abort(-1);
57:     }
58:     else
59:         printf("command file is %s\n", argv[1]);
60:
61:     fp = fopen(argv[1], "r");
62:     if (fp == NULL) {
63:         printf("could not open command file\n");
64:         ADLB_Abort(-1);
65:     }
66:
67:     while (fgets(cmdbuffer, 1024, fp) != NULL) {
68:         cmdbuffer[strlen(cmdbuffer)] = '\0';
69:         printf("command = %s", cmdbuffer); /* cmd includes \n */
70:         if (cmdbuffer[0] != '#') {
71:             /* put command into adlb here */
72:             rc = ADLB_Put( cmdbuffer, strlen(cmdbuffer)+1, -1, -1,
73:                 CMDLINE, 1 );
74:             aprintf( 1, "put cmd, rc = %d\n", rc );
75:         }
76:     }
77:     printf("\nall commands submitted\n");
78: }
```

```
79:      /* all application processes, including the application master,
80:      execute this loop */
81:
82:     done = 0;
83:     while ( !done ) {
84:         req_types[0] = -1;
85:         req_types[1] = req_types[2] = req_types[3] = -1;
86:         aprntf( 1, "Getting a command\n" );
87:         rc = ADLB_Reserve( req_types, &work_type, &work_prio,
88:                          work_handle, &work_len, &answer_rank);
89:         /* (work_handle is an array, so no & in above call) */
90:         aprntf( 1, "rc from getting command = %d\n", rc );
91:         if ( rc == ADLB_DONE_BY_EXHAUSTION ) {
92:             aprntf( 1, "All jobs done\n" );
93:             break;
94:         }
95:         if ( rc == ADLB_NO_MORE_WORK ) {
96:             aprntf( 1, "No more work on reserve\n" );
97:             break;
98:         }
99:         else if ( rc < 0 ) {
100:             aprntf( 1, "Reserve failed, rc = %d\n", rc );
101:             ADLB_Abort(-1);
102:         }
103:         else if ( work_type != CMDLINE) {
104:             aprntf( 1, "unexpected work type %d\n", work_type );
105:             ADLB_Abort( 99 );
106:         }
```

```
107:         else {                               /* reserved good work */
108:             rc = ADLB_Get_reserved( cmdbuffer, work_handle );
109:             if (rc == ADLB_NO_MORE_WORK) {
110:                 aprprintf( 1, "No more work on get_reserved\n" );
111:                 break;
112:             }
113:             else {                               /* got good work */
114:                 /* print command to be executed */
115:                 /* printf("executing command line :%s:\n", cmdbuffer); */
116:                 system( cmdbuffer );
117:             }
118:         }
119:     }
120: }
121: ADLB_Finalize();
122: MPI_Finalize();
123:
124: return(0);
125: }
```