```
// server.c

/*******************************************************************************
* Process Management Server: Child Server Process
* Author: Michael Hartung
* Date: 2/12/2015
* Description: This process begins when execl is called from the master server
* and is responsible for handling it's own child processes.
*******************************************************************************/

#include "server.h"

int minProcesses;
int maxProcesses;
int numActive;
pid_t pid[MAX_PROCESSES];
char* serverName;
int sigIntReceived;

int main(int argc, char* argv[])
{
          serverName = argv[2];
          sigIntReceived = 0;
          //Register Signals
          signal(SIGINT, sigIntHandler);
          signal(SIGUSR1, sigUSR1Handler);
          signal(SIGUSR2, sigUSR2Handler);
          numActive = 0;
          minProcesses = atoi(argv[0]);
          maxProcesses = atoi(argv[1]);
          int i = 0;

          // Creates processes until at minProcesses
          for(i = 0; i < minProcesses; i++)
          {
                    createProcess();
          }

          // Waits for signals
          while(1);

          return 0;
}


// Handles kill command from master server
void sigIntHandler(int sigNum)
{
          sigIntReceived = 1;
          while(numActive > 0)
          {
                    abortProcess();
                    numActive--;
          }
          printf("%s exiting...\n", serverName);
          exit(0);
}

// Sends SIGUSR1 as a command to create a process
void sigUSR1Handler(int sigNum)
{
          createProcess();
}

// Sends SIGUSR2 as a command to abort a process
void sigUSR2Handler(int sigNum)
```

```
{
          abortProcess();
}

// Creates a process on server if not already at indicated maximum
void createProcess()
{
          if(numActive < maxProcesses)
          {
                    if((pid[numActive] = fork()) < 0)
                    {
                              printf("Fork Broke!\n");
                    }
                    else if(!pid[numActive])
                    {
                              execl("process", NULL);
                    }
                    numActive++;
          }
          else
          {
                    printf("Too many processes running on %s\n", serverName);
          }
}

// Aborts a process if not already at indicated minimum
void abortProcess()
{
          if(numActive > minProcesses || sigIntReceived)
          {
                    if(numActive > 0)
                    {
                              numActive--;
                              kill(pid[numActive], SIGINT);
                    }
                    else
                    {
                              printf("%s: No Processes to terminate!\n", serverName);
                    }
          }
          else
          {
                    printf("Too few processes in %s to abort!\n", serverName);
          }
}
```