

```

#include <unistd.h>
#include "measure.h"

int running = 1;
params_t *p; //pointer to shared memory

void main (void)
{
    create shared memory space;
    switch (fork())
    {
        case -1:
            printf ("fork failed\n");
            break;

        case 0: // child process
            attach to shared memory space;
            while (running)
            {
                fgets();
                parse command;
                put result in shared memory;
            }
            break;

        default: // parent process
            attach to shared memory space;
            while (running)
            {
                read A/D;
                act on current state;
            }
            break;
    }
    exit (0);
}

```

**Figure 1: fork()**

```

#include <unistd.h>
#include "measure.h"

int running = 1;
params_t *p; //pointer to shared memory

void main (void)
{
    create shared memory space;
    switch (fork())
    {
        case -1:
            printf ("fork failed\n");
            break;

        case 0: // child process
            Put p into argv for execve
            execve ("monitor", argv, environ);
            printf ("execve failed\n");
            break;

        default:
            while (running)
            {
                read A/D;
                act on current state;
            }
            break;
    }
    exit (0);
}

```

**Figure 2: execve()**

```
/*      Thread Example  */
#include <pthread.h>
#include "errors.h"

/*
 * Thread start routine.
 */
void *thread_routine (void *arg)
{
    printf ("%s\n", arg);
    return arg;
}

main (int argc, char *argv[])
{
    pthread_t thread_id;
    void *thread_result;
    int status;

    status = pthread_create (&thread_id, NULL,
        thread_routine, (void *)argv[1]);
    printf ("New thread created\n");

    status = pthread_join (thread_id,
    &thread_result);
    printf ("Thread returned %p\n", thread_result);
    return 0;
}
```

**Figure 3: thread example**