Real Time Systems Assignment 1.1

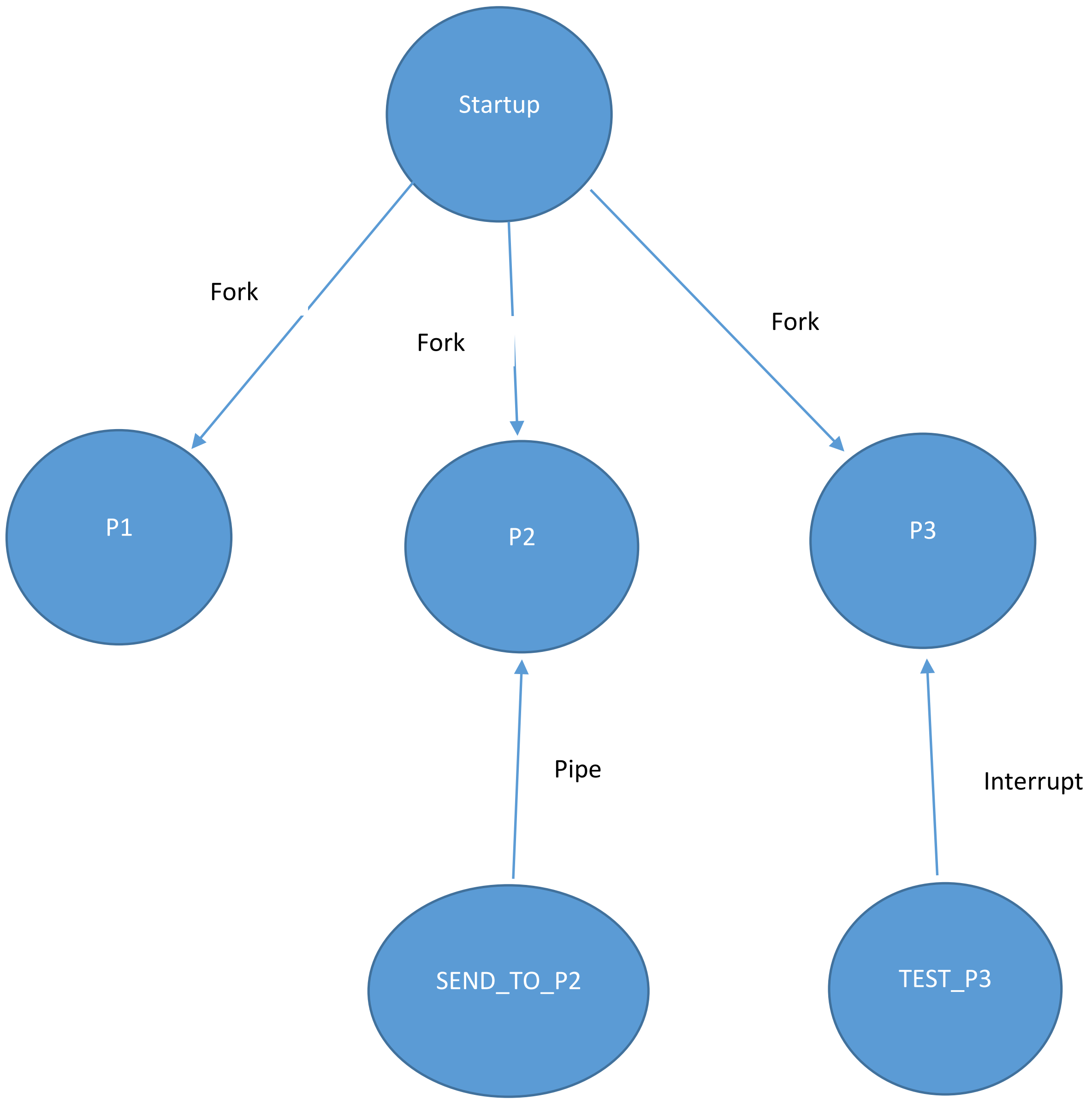
CORK INSTITUTE OF TECHNOLOGY

INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Name: Michael O' Sullivan
Date: 21/10/16
Class: DCOM4

Overview

For the first assignment of Real Time System we have to create a startup process with 3 child processes. The startup process forks each of the child process to initiate them. This is a diagram showing the processes.



Problem 1.1

Startup

This c file executes a fork, this fork starts processes P1, P2 and P3. _os_exec() prepares the parameter and environment list before creating a process. _os_fork() creates a new process which becomes a child of the caller. It sets up the new process' memory, MPU registers, and standard I/O paths. For the P1 fork it is passing in an string array. This array will be printed from P1.

Code

```
#include <stdio.h>
#include <errno.h>
#include <process.h>
#include <dexec.h>
#include <types.h>
#include <string.h>
#include <modes.h>
#include "CommData.h"

main(int argc , char * argv[], char **envp)
	{
	error_code err;
	error_code errP2;
	error_code errP3;

	process_id p1_pid;
	process_id p2_pid;
	process_id p3_pid;

	status_code child_status;
	status_code child_status2;
	status_code child_status3;


	path_id pipe_P2;
	u_int32 count;
	u_int32 pipe_size1;
	struct CommData MyData;

char * p1_argv[] = {
			"P1",
			"A real-time operating system (RTOS) is an operating system that guarantees a certain capability within a specified time constraint.",
			"For example, an operating system might be designed to ensure that a certain object was available for a robot on an assembly line.",
			"In what is usually called a hard real-time operating system, if the calculation could not be performed for making the object available at the designated time, the operating system would terminate with a failure. ",
                        "In a soft real-time operating system, the assembly line would continue to function
but the production output might be lower as objects failed to appear at their designated time,
causing the robot to be temporarily unproductive. ",
                        "Some real-time operating systems are created for a special application and others
are more general purpose. "
};
/* ---------------P1-------------------- */
/* fork p1 process */
if (err = (_os_exec(_os_fork, 0 , 3 , p1_argv[0], p1_argv, envp,0,&p1_pid, 0,0) != 0))
            printf("Error1\n");
/* ---------------P2-------------------- */
 /* fork p2 process */

 if (err = (_os_exec(_os_fork, 0 , 3 , "P2", p1_argv, envp,0,&p2_pid, 0,0) != 0))
            printf("Error1\n");
/* ---------------P3-------------------- */

/* fork p3 process */
if (err = (_os_exec(_os_fork, 0 , 3 ,"P3", p1_argv, envp,0,&p3_pid, 0,0) != 0))
            printf("Error1\n");

/* wait for child P1 */
if (err = (_os_wait(&p1_pid, &child_status) != 0))
            printf("Error1\n");

/* wait for child p2 */
if (err = (_os_wait(&p2_pid, &child_status2) != 0))
            printf("Error2\n");


/* wait for child  p3*/
if (err = (_os_wait(&p3_pid, &child_status3) != 0))
            printf("Error3\n");




}
```

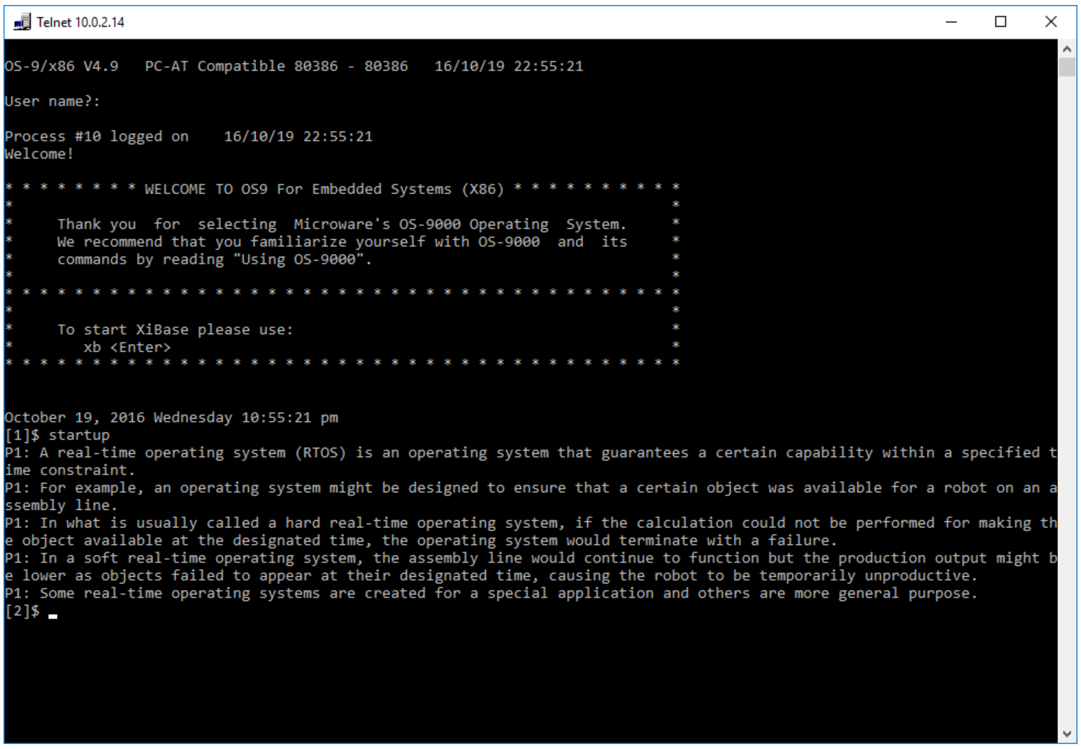
P1

P1 accepts the array from the startup process. Using a for loop it will print out each array index. After it prints an index of the array the system will wait 4.2 seconds before it prints another index of the array. This is achieved by using _os_sleep. _os_sleep() deactivates the calling process until the requested number of ticks have elapsed. For this process the number of ticks is 4200(4.2 seconds).

Code

```
#include <stdio.h>
#include <errno.h>
#include <process.h>
#include <types.h>

main(int argc , char * argv[])
        {
        u_int32 numTicks;
        signal_code dummySignal;
        int a;

          /* for loop execution */
          for( a =1 ; a < 6; a = a + 1 ){
                  numTicks = 4200;
            printf("P1:    %s\n",argv[a]);
            _os_sleep(&numTicks, &dummySignal);
            }

        }
```

Output



```
OS-9/x86 V4.9   PC-AT Compatible 80386 - 80386   16/10/19 22:55:21

User name?:

Process #10 logged on    16/10/19 22:55:21
Welcome!

* * * * * * * * WELCOME TO OS9 For Embedded Systems (X86) * * * * * * * * * * *
*                                                                            *
*     Thank you  for  selecting  Microware's OS-9000 Operating  System.      *
*     We recommend that you familiarize yourself with OS-9000  and  its      *
*     commands by reading "Using OS-9000".                                   *
*                                                                            *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                            *
*     To start XiBase please use:                                            *
*        xb <Enter>                                                          *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *


October 19, 2016 Wednesday 10:55:21 pm
[1]$ startup
P1: A real-time operating system (RTOS) is an operating system that guarantees a certain capability within a specified t
ime constraint.
P1: For example, an operating system might be designed to ensure that a certain object was available for a robot on an a
ssembly line.
P1: In what is usually called a hard real-time operating system, if the calculation could not be performed for making th
e object available at the designated time, the operating system would terminate with a failure.
P1: In a soft real-time operating system, the assembly line would continue to function but the production output might b
e lower as objects failed to appear at their designated time, causing the robot to be temporarily unproductive.
P1: Some real-time operating systems are created for a special application and others are more general purpose.
[2]$ _
```

Status: 100%

P2

For this process, it will wait for 20 seconds and then check a pipe to see if there is a message available. This message will be sent from a process call “send_to_p2”. Once the process is started from the startup process, _os_sleep will be executed. P2 uses _os_open open a path and see if there is a pipe in there. If there is a message there _os_read() reads the number of bytes from the specified path. The path must previously have been opened in read or update mode. The message is then printed.

Code

```
#include <stdio.h>
#include <errno.h>
#include <modes.h>
#include <signal.h>
#include <types.h>
#include "CommData.h"

main()
        {
        u_int32 numTicks;
        signal_code dummySignal;
        error_code err;
        path_id pipe_P2;
        u_int32 count;
        struct CommData MyData;


        numTicks = 20000;
        _os_sleep(&numTicks, &dummySignal);

        if ((err = _os_open("/pipe/P2", S_IREAD, &pipe_P2)) != 0)
                {
                printf("P2:     No Message availible. \n");
                exit(0);
                }
        count = MESSAGE_SIZE;
        if ((err = _os_read(pipe_P2, &MyData, &count)) !=0)
                {
                printf("Error : Message reading error invalid message?");
                exit(0);
                }
        else
                {
                printf("Message Received\n");
                printf("P2:             Sensor: %s\n", MyData.From);
                printf("P2:             MessageNumber: %d\n", MyData.MessageNumber);
                printf("P2:             Water Info = %s'\n", MyData.Message);
                printf("P2:             Temperature = %d c\n", MyData.Temp);
                printf("P2:             Pressure = %d\n", MyData.Pressure);
```

```
			printf("P2:			Valve open:  = %s\n", MyData.ValveOpen);
			printf("P2:			Water Level:  = %d\n", MyData.WaterLevel)

			}
		_os_close(pipe_P2);
}
```

Send_to_p2

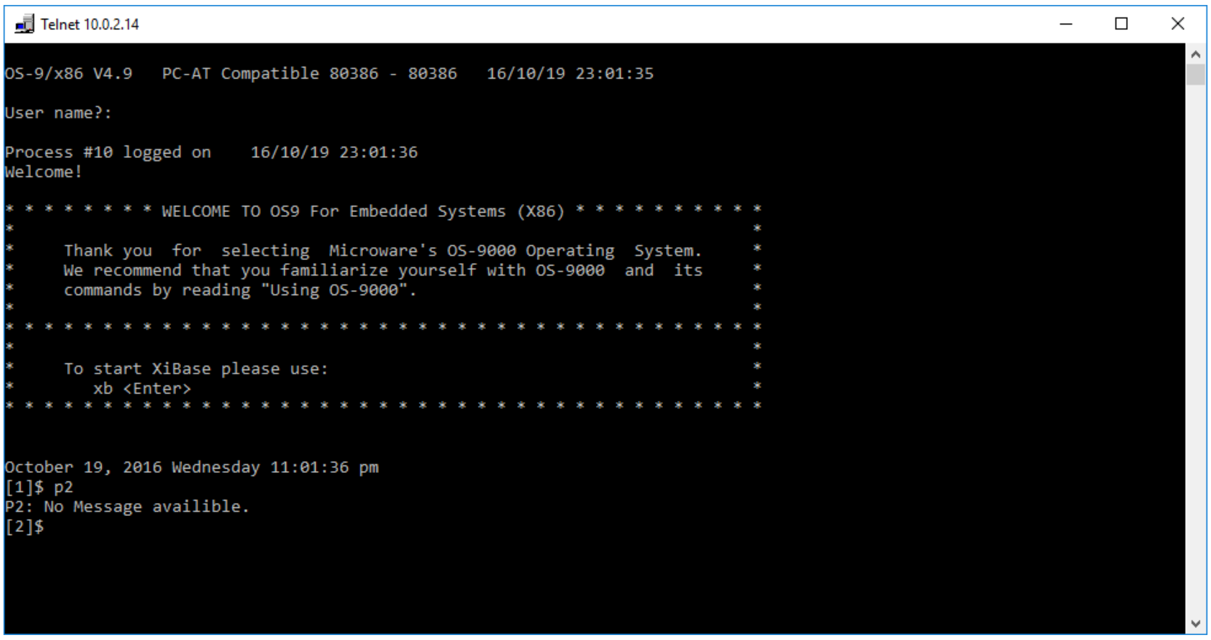
This process is used to create a new file using os_create which will contain the data from CommData.h and store it in a path.

Code

```
#include <stdio.h>
#include <errno.h>
#include <modes.h>
#include <types.h>
#include <string.h>
#include "CommData.h"

main()
	{
	error_code err;
	path_id pipe_P2;
	u_int32 count;
	u_int32 pipe_size1;
	struct CommData MyData;

	if ((err = _os_create ("/pipe/p2", S_IREAD | S_ISIZE, &pipe_P2, FAP_READ | FAP_GREAD |
FAP_WRITE | FAP_GWRITE, 501)) !=0)
	{

		printf("The pipe for the process P2 has a message ready? \n");

		count = MESSAGE_SIZE;
		if (( err= _os_open("/pipe/p2", S_IWRITE, &pipe_P2)) !=0)
		{
			printf("Error -- Cannot open a pipe to send to????\n");
			exit(0);
		}
	}

strcpy(MyData.From,"P0");
MyData.MessageNumber = 2001;
strcpy(MyData.Message, "Water sensor");
MyData.Temp = 34;
MyData.Pressure = 51;
strcpy(MyData.ValveOpen,"Y");
MyData.WaterLevel = 20;
count = MESSAGE_SIZE;
if ((err = _os_write(pipe_P2, &MyData, &count)) != 0)
        printf("Error: write...");

-_os_close(pipe_P2);
}
```

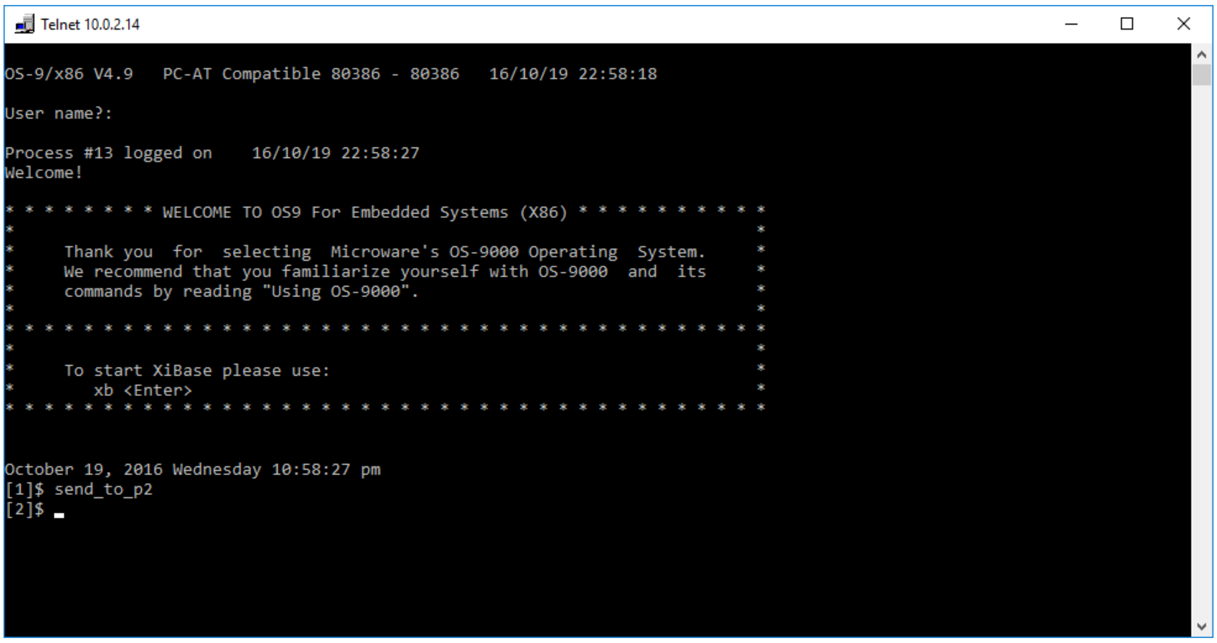
Output

The first output is without any message being sent to P2.

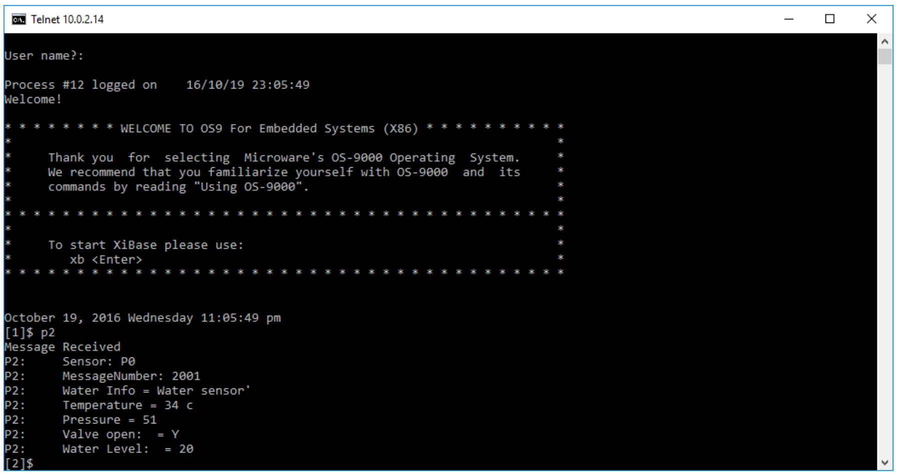


```
OS-9/x86 V4.9   PC-AT Compatible 80386 - 80386   16/10/19 23:01:35

User name?:

Process #10 logged on    16/10/19 23:01:36
Welcome!

* * * * * * * * WELCOME TO OS9 For Embedded Systems (X86) * * * * * * * * * *
*                                                                            *
*     Thank you  for  selecting  Microware's OS-9000 Operating  System.      *
*     We recommend that you familiarize yourself with OS-9000  and  its      *
*     commands by reading "Using OS-9000".                                   *
*                                                                            *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 
*                                                                            *
*     To start XiBase please use:                                            *
*        xb <Enter>                                                          *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 


October 19, 2016 Wednesday 11:01:36 pm
[1]$ p2
P2: No Message availible.
[2]$
```

I will now send a message through a pipe to P2 from send_to_p2



```
OS-9/x86 V4.9   PC-AT Compatible 80386 - 80386   16/10/19 22:58:18

User name?:

Process #13 logged on    16/10/19 22:58:27
Welcome!

* * * * * * * * WELCOME TO OS9 For Embedded Systems (X86) * * * * * * * * * *
*                                                                            *
*     Thank you  for  selecting  Microware's OS-9000 Operating  System.      *
*     We recommend that you familiarize yourself with OS-9000  and  its      *
*     commands by reading "Using OS-9000".                                   *
*                                                                            *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 
*                                                                            *
*     To start XiBase please use:                                            *
*        xb <Enter>                                                          *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 


October 19, 2016 Wednesday 10:58:27 pm
[1]$ send_to_p2
[2]$ _
```



```
User name?:

Process #12 logged on    16/10/19 23:05:49
Welcome!

* * * * * * * * WELCOME TO OS9 For Embedded Systems (X86) * * * * * * * * * *
*                                                                           *
*     Thank you  for  selecting  Microware's OS-9000 Operating  System.     *
*     We recommend that you familiarize yourself with OS-9000  and  its     *
*     commands by reading "Using OS-9000".                                  *
*                                                                           *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                           *
*     To start XiBase please use:                                           *
*        xb <Enter>                                                         *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *


October 19, 2016 Wednesday 11:05:49 pm
[1]$ p2
Message Received
P2:     Sensor: P0
P2:     MessageNumber: 2001
P2:     Water Info = Water sensor'
P2:     Temperature = 34 c
P2:     Pressure = 51
P2:     Valve open:  = Y
P2:     Water Level:  = 20
[2]$
```

Status: 100%

P3

This process waits for interrupt signal messages send from test_p3.

Code

```
#include <stdio.h>
#include <signal.h>
#include <errno.h>
#include <modes.h>
#include <types.h>
#include <cglob.h>

sig_handler(signal_code sig)
        {

        switch (sig)
                {
                case 400 : printf("P3:    Case 400\n");
                break;

                case 500 : printf("P3:    Case 500\n");
                break;

                case 600 : printf("P3:    Case 600\n");
                break;

                case 700 : printf("P3:    Case 700\n");
                break;
          }

        _os_rte();
        }
main()
        {
         error_code err;
         u_int32 num_ticks;
         signal_code dummy_sig;

        if ((err = _os_intercept(sig_handler, _glob_data)) != 0)
                exit(err);
                num_ticks = 40000;
                _os_sleep(&num_ticks, &dummy_sig);


}
```

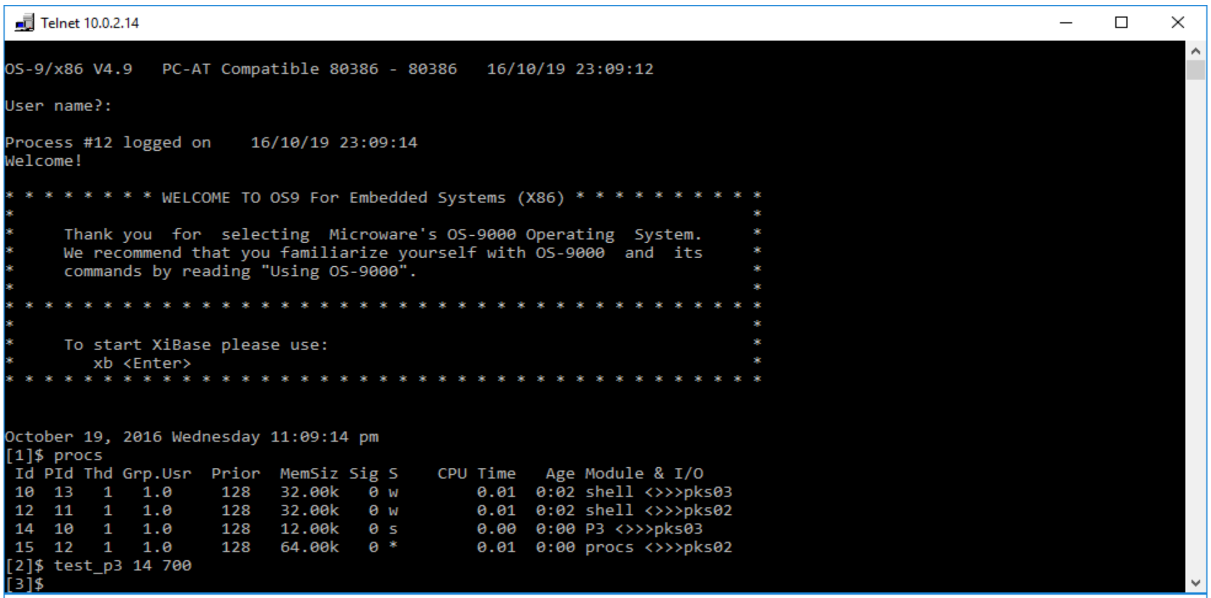
Test_P3 Code

This process sends an interrupt to P3 using _os_send. _os_send() sends a signal to the specified process. A process may send the same signal to all processes of the same group/user ID by passing zero as the receiving process' ID number. To get the process ID of P3, enter procs on the command line and it will give you all the running processes.

Code

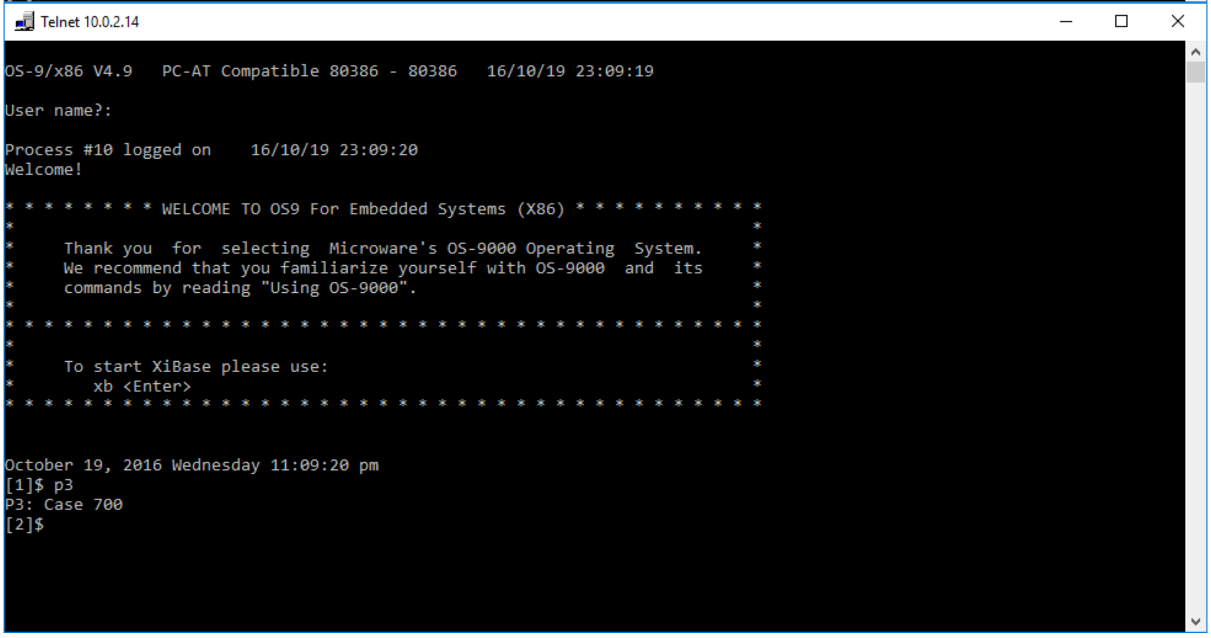
```
#include <stdio.h>
#include <signal.h>
#include <types.h>

main (int argc, char * argv[])
        {
        _os_send(atoi(argv[1]), atoi(argv[2]));
        }
```

Outputs



Telnet 10.0.2.14

```
OS-9/x86 V4.9   PC-AT Compatible 80386 - 80386   16/10/19 23:09:12

User name?:

Process #12 logged on    16/10/19 23:09:14
Welcome!

* * * * * * * * WELCOME TO OS9 For Embedded Systems (X86) * * * * * * * * * *
*                                                                            *
*     Thank you  for  selecting  Microware's OS-9000 Operating  System.      *
*     We recommend that you familiarize yourself with OS-9000  and  its      *
*     commands by reading "Using OS-9000".                                   *
*                                                                            *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                            *
*     To start XiBase please use:                                            *
*        xb <Enter>                                                          *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *


October 19, 2016 Wednesday 11:09:14 pm
[1]$ procs
 Id PId Thd Grp.Usr  Prior  MemSiz Sig S    CPU Time   Age Module & I/O
 10  13   1   1.0     128   32.00k   0 w        0.01  0:02 shell <>>>pks03
 12  11   1   1.0     128   32.00k   0 w        0.01  0:02 shell <>>>pks02
 14  10   1   1.0     128   12.00k   0 s        0.00  0:00 P3 <>>>pks03
 15  12   1   1.0     128   64.00k   0 *        0.01  0:00 procs <>>>pks02
[2]$ test_p3 14 700
[3]$
```



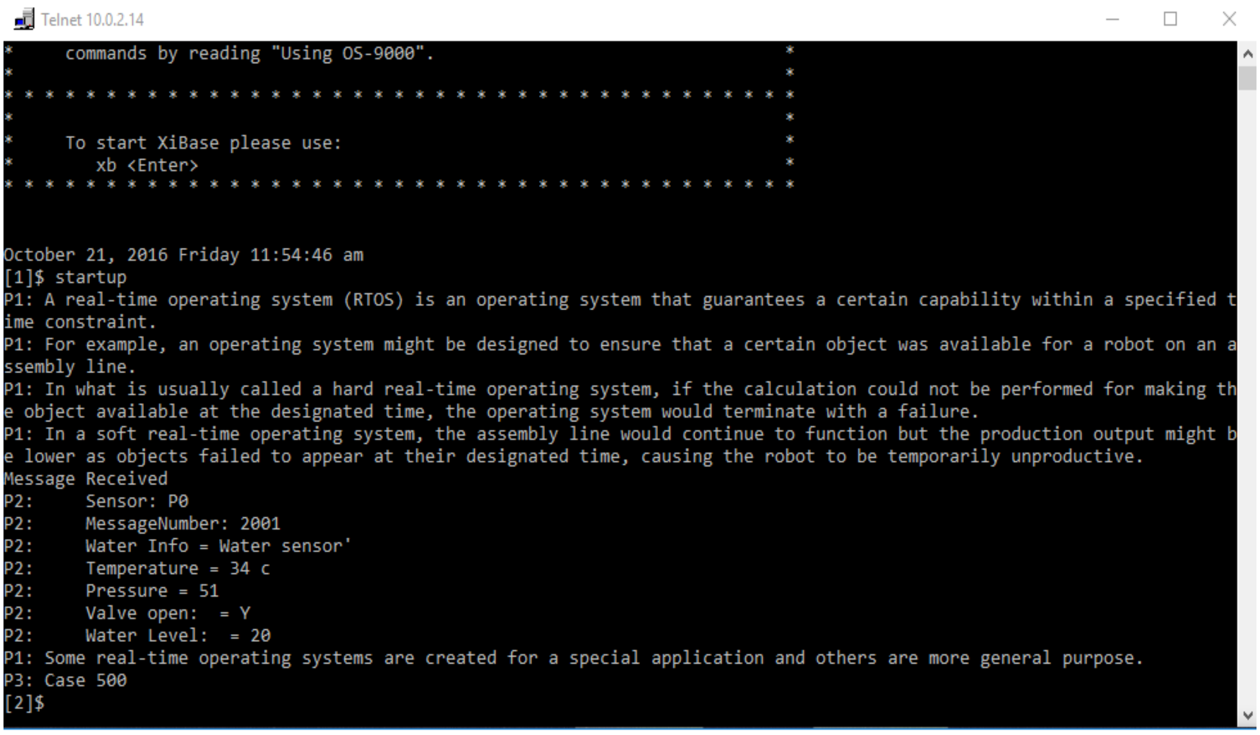
Telnet 10.0.2.14

```
OS-9/x86 V4.9   PC-AT Compatible 80386 - 80386   16/10/19 23:09:19

User name?:

Process #10 logged on    16/10/19 23:09:20
Welcome!

* * * * * * * * WELCOME TO OS9 For Embedded Systems (X86) * * * * * * * * * *
*                                                                            *
*     Thank you  for  selecting  Microware's OS-9000 Operating  System.      *
*     We recommend that you familiarize yourself with OS-9000  and  its      *
*     commands by reading "Using OS-9000".                                   *
*                                                                            *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                            *
*     To start XiBase please use:                                            *
*        xb <Enter>                                                          *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *


October 19, 2016 Wednesday 11:09:20 pm
[1]$ p3
P3: Case 700
[2]$
```

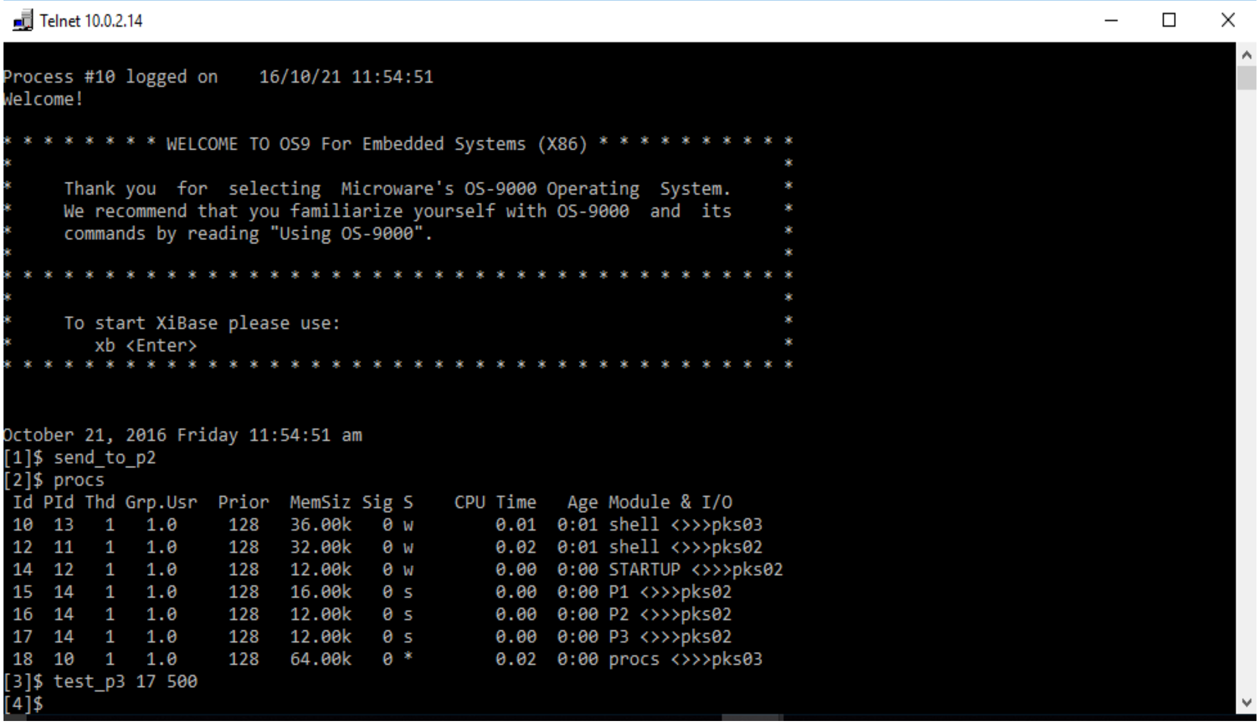
Status: 100%

Concurrent Processes

This is two telnets running side by side. I first sent a message to the pipe using the bottom telnet. When the startup ran, it out putted P1, after 20 seconds P2 looks for a message, P2 then displays the message sent by send_to_p2. To test P3 I found the PId using the "procs" command then using test_p3 I put in the arguments of P3 PId and the signal message.



```
*     commands by reading "Using OS-9000".                               *
*                                                                        *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 
*                                                                        *
*     To start XiBase please use:                                        *
*        xb <Enter>                                                      *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 


October 21, 2016 Friday 11:54:46 am
[1]$ startup
P1: A real-time operating system (RTOS) is an operating system that guarantees a certain capability within a specified t
ime constraint.
P1: For example, an operating system might be designed to ensure that a certain object was available for a robot on an a
ssembly line.
P1: In what is usually called a hard real-time operating system, if the calculation could not be performed for making th
e object available at the designated time, the operating system would terminate with a failure.
P1: In a soft real-time operating system, the assembly line would continue to function but the production output might b
e lower as objects failed to appear at their designated time, causing the robot to be temporarily unproductive.
Message Received
P2:     Sensor: P0
P2:     MessageNumber: 2001
P2:     Water Info = Water sensor'
P2:     Temperature = 34 c
P2:     Pressure = 51
P2:     Valve open:  = Y
P2:     Water Level:  = 20
P1: Some real-time operating systems are created for a special application and others are more general purpose.
P3: Case 500
[2]$
```



```
Process #10 logged on    16/10/21 11:54:51
Welcome!

* * * * * * * * WELCOME TO OS9 For Embedded Systems (X86) * * * * * * * * * * 
*                                                                        *
*     Thank you  for  selecting  Microware's OS-9000 Operating  System.  *
*     We recommend that you familiarize yourself with OS-9000  and  its  *
*     commands by reading "Using OS-9000".                               *
*                                                                        *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 
*                                                                        *
*     To start XiBase please use:                                        *
*        xb <Enter>                                                      *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 


October 21, 2016 Friday 11:54:51 am
[1]$ send_to_p2
[2]$ procs
 Id PId Thd Grp.Usr  Prior  MemSiz Sig S    CPU Time   Age Module & I/O
 10  13   1   1.0     128  36.00k   0 w        0.01  0:01 shell <>>>pks03
 12  11   1   1.0     128  32.00k   0 w        0.02  0:01 shell <>>>pks02
 14  12   1   1.0     128  12.00k   0 w        0.00  0:00 STARTUP <>>>pks02
 15  14   1   1.0     128  16.00k   0 s        0.00  0:00 P1 <>>>pks02
 16  14   1   1.0     128  12.00k   0 s        0.00  0:00 P2 <>>>pks02
 17  14   1   1.0     128  12.00k   0 s        0.00  0:00 P3 <>>>pks02
 18  10   1   1.0     128  64.00k   0 *        0.02  0:00 procs <>>>pks03
[3]$ test_p3 17 500
[4]$
```

Status: 100%

Declaration of work

This report has been constructed and produced by Michael O' Sullivan. I declare that the report and its contents have been produced by Michael O Sullivan and is entirely his own work.

Mr. Michael O’Sullivan

R00077764

X_____Michael O Sullivan________________

Date: __21_____/ 10_____/ 16______