



MP3007

Date Nov 2008

No.

- Q1a)
- line 1: declare local string variable s_1 & s_2
 - line 2: assign each character in s_2 to s_1 until the end of the string where '\0' is placed. once '\0' is assigned, while loop is terminated.
 - line 3: assign characters again
 - line 4: pointer moves to the next character of receiving string
 - line 5: pointer moves to the next character of sending string
 - line 6: end of while loop
 - line 7: return the receiving string's last character's address

s_1 & s_2 are both pointers. $s_1 = s_2$ is to make s_1 point to the same address as s_2 , but the content of s_2 is not copied

```
b) strcpy (s2, s1)
printf ("string 1 is : \n");
while (*s1);
{
    printf ("%s", s1);
    s1++;
}
printf ("string 2 is : \n");
while (*s1);
{
    printf ("%s", s2);
    s2++;
}
```

```
c) char * strcpy (char *s1, char *s2)
{
    while (*s2)
    {
        if ((*s2 >= 65) && (*s2 <= 90))
            *s1 = *s2 + 32;
        else
            *s1 = *s2;
    }
    return s1;
}
```

Q 200

port I/O

simple timing needs
slower

no memory reduction

memory mapped I/O

complex

fast

reduce memory address

structure of program.

- ① system asks permission to access the memory
- ② system maps a memory location to the port.
- ③ system takes the reading from the port, clear the redundant higher 4 bits and store it to 'data'.
- ④ system clears the higher 4 bits of data and output it back to the port.

'Thread Ctrl' is to ask permission to access the memory

'mmap_device_io' is assign a memory address to the port

'in8()' is to take a 8-bit reading from the specified memory location

'out8()' is to output a 8-bit data to the specified memory location.

In RTOS, many users can use the system simultaneously, in order to prevent the memory from being randomly altered, the system has to place a control to regulated the accesses

```
while (data)
```

```
{ int n=0;
```

```
    if (data%2)
```

```
        printf("bit %d is 1", n);
```

```
    else
```

```
        printf("bit %d is 0", n);
```

```
    data >> 1;
```

```
    n++;
```

```
}
```

// check if current 'data' is even

// move 2nd lowest bit to the LSB for the next checking



```
int main()
{
    int data=0, data1=0;
    uintptr_t iobase;

    ThreadCtl(-NTD_TCTL_IO, NULL);
    iobase = mmap_device_io(16, 0x300);

    data = in8(iobase+3) & 0x0f;

    while(data)
    {
        int n=0;
        if (data%2)
            printf("bit %d is 1", n);
        else
            printf("bit %d is 0", n); data1 += 2^n; } // complemented data is
        data >> 1; // stored in data1
        n++;
    }

    out8(iobase+3, data1 & 0x0f);
    exit(0);
}
```



Date

No.

Q3a) A separate process running continuously to poll the status of buttons, it works as a 'client' to the main process. Once a button is pushed, this process will send a `MsgSend()` to the main process. In `MsgSend()`, the respective option will also be included so that the main process will carry out different operations accordingly.

```
b) #include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
```

```
int temp = 0;
int main() {
    dog_into_oven();
    Thread Ctrl (-NTO-TCTL-IO, 0);
    tobse = mmap-device-io(16, 0x300);
```

```
do
{
    out8(base, 0x00);
    delay(5);
    while (in8(base+8) < 0x80);
    {
        temp = in8(base+1);
        if (temp <= 160)
            sleep(1);
    }
} while (temp <= 160)
serve_dog();
}
```



- c) we can use timer in this case, is linked
- ① declare a timer variable: `timer_t tid;`
 - ② declare a timer specification variable: `itimerspec t;`
 - ③ declare a structure which links signal to event: `struct sigevent evp;`
 - ④ assign SIGUSR1 as the signal to trigger the event: `evp.sigev signo = SIGUSR1;`
 - ⑤ change the property of timer.
 - `t.it_value.tv_sec = 0;`
 - `t.it_value.tv_nsec = 1; // set starting time to be 1 nsec after triggering`
 - `t.it_interval.tv_sec = 200;`
 - `t.it_interval.tv_nsec = 0; // set period to be 5 min`
 - ⑥ `evp.sigev_notify = SIGEV_SIGNAL;`
 - ⑦ use function `test_status()` as the signal handler for SIGUSR1:
`signal(SIGUSR1, test_status);`
 - ⑧ create the timer: `timer_create(CLOCK_REALTIME, &evp, &tid);`
 - ⑨ start the time: `timer_settime(&tid, 0, &t, NULL);`

	mode	screen output	Date	No.
84 a)	P_WAIT.	A B C		
	P_NOWAIT.	A B C or A C B		
	P_OVERLAY	A B		

b) # main process #

```
int main()
{
    pid_t child;
    child = spawnl(P_NOWAIT, "geton", NULL);
    getchar();
    kill(child, SIGTERM);
}
```

process with filename "geton"

```
int main()
{
    while(1)
    {
        printf("Get on with it!");
        sleep(5);
    }
}
```

hence, by press any key in the main process, the child process will be terminated

c). Scheduling is a way to allocate time among different processes. In this way, processes will run concurrently and priority could be given to more important processes.



```
d) int main()
{
    signal(SIGINT, printmsg);
    while(1)
        pause;

    static void printmsg()
    {
        delay(10);
        printf("Hello World!");
    }
}
```

吴禹力

quality assured