Assignment 2 Report

- **Does your program output any garbage? If yes, why?**

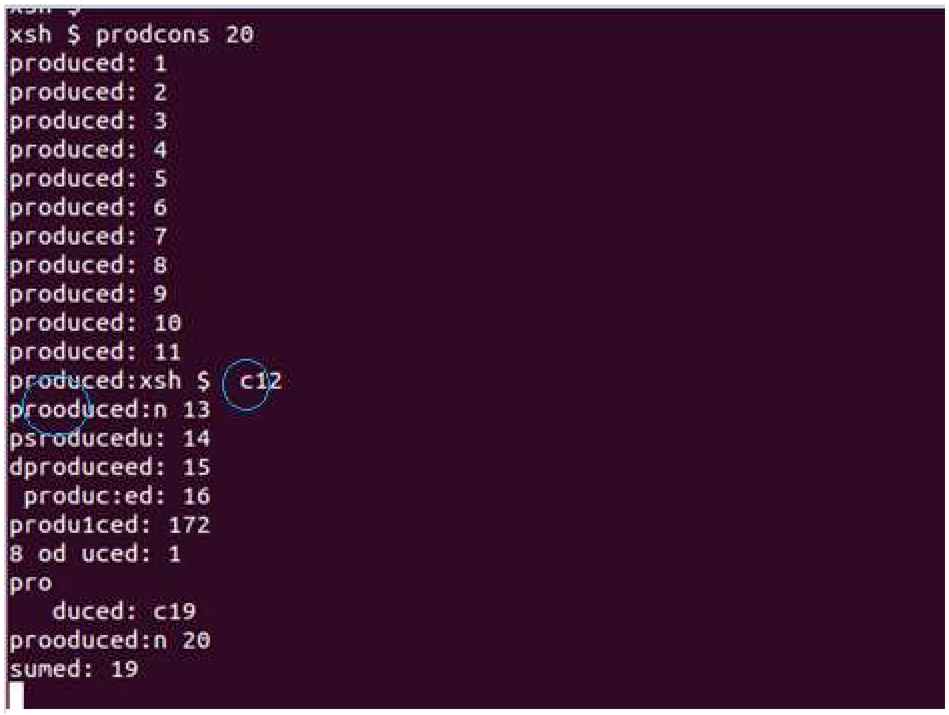
Yes, the program prints garbage values/ jumbled characters on to the terminal.

1. It looks like the output of producer and consumer are getting printed on the console simultaneously. This is because:
2. When the producer is executing the print statement, it puts the characters in the output buffer to print on console.
3. However, before it can do so, the context switch occurs and now the consumer process starts putting its character into the output buffer.
4. As a result, the console displays the jumbled characters from both producer and consumer process.

This could be fixed if there is a way to synchronize the two processes to use the output buffer atomically.

Example:

Fig 1.1: prodcons shell command output for count = 20



```
xsh $ prodcons 20
produced: 1
produced: 2
produced: 3
produced: 4
produced: 5
produced: 6
produced: 7
produced: 8
produced: 9
produced: 10
produced: 11
produced:xsh $  c12
prooduced:n 13
psroducedu: 14
dproduceed: 15
 produc:ed: 16
produ1ced: 172
8 od uced: 1
pro
   duced: c19
prooduced:n 20
sumed: 19
```

In Figure 1.1,

1. We can see that as producer produces value 12, the consumer process begins and starts putting its output in the console buffer. (Refer 2nd blue highlight in fig 1.1)
2. However, before the consumer process can completely print “consumer”, a context switch occurs once “C” is printed on the console and the control is given back to the producer process.
3. The producer process then starts printing for produced value 13, however, like before, a context switch occurs once it outputs “pro” and a character “o” from the consumer process is printed on the console. (Refer 2nd blue highlight in fig 1.1)
4. Thus, we can see jumbled or garbage characters, whenever there is a context switch between the two processes using shared resources (console).

- **Are all the produced values getting consumed? Check your program for a small count like 20.**

No. All produced values are not consumed. Following are few observations for this issue-

1. First, the producer starts execution and by the time, consumer process is ready to execute, producer has already produced few values and assigned to n.
2. Now, when consumer consumes ‘n’, it will get the current value of n and all previous values are skipped.
3. When producer completes its execution for producing all values of ‘n’ until it reaches count, the consumer is still in middle of its execution.
4. Consumer executes print statement to print value stored in its output buffer.
5. Now, Consumer checks whether n has reached ‘count’, then, it stops its execution.
6. We note that few of the last produced values are missed by consumer for printing.

Example:

In figure 1.1 above, we observe execution steps explained below -

1. Producer has produced values from 1 to 12.
2. Now, consumer starts its execution and starts printing from 12.It missed all previous values.

3. When producer has completed producing values, consumer is still executing its second statement to check whether ‘n’ has reached count.
4. Consumer again goes to print value of n which it gets as ’19’ since producer is continuously producing values.
5. And by the time, it reaches to compare n with count, n has already reached its maximum value – 20. Hence, consumer stops it execution and misses few of last produced values.

Source Code Changes:

prodcons.h

```
#include <xinu.h>
#include <stddef.h>
 #include <stdio.h>

/*Global variable for producer consumer*/
extern int n; /*this is just declaration*/

/*function Prototype*/
void consumer(int count);
void producer(int count);
```

produce.c

```
#include <prodcons.h>

 void producer(int count)
 {
     //Code to produce values less than equal to count,
            int i;
            for(i = 1; i <= count; i++)
            {
                    n = i;
                    printf("produced: %d \n",n);
            }
 }
```

consume.c

```
#include <prodcons.h>
void consumer(int count)
{
        while (1){
                printf("consumed: %d \n",n);
                if ( n == count){
                        break;
                }
        }
}
```

xsh_prodcons.c

```
#include <prodcons.h>
#include <ctype.h>
int n ;             //Definition for global variable 'n'
/*Now global variable n will be on Heap so it is accessible all the processes i.e. consume
and produce*/

shellcmd xsh_prodcons(int nargs, char *args[])
{
     //Argument verifications and validations

     int count = 2000;        //local varible to hold count
        int i = 0;

        // Initialise the value of n to 0, since this is an extern variable, it may start with
the previous value
        n = 0;
        /* Output info for '--help' argument */
        if (nargs == 2 && strncmp(args[1], "--help", 7) == 0)
        {
                printf("Usage: %s\n\n", args[0]);
                printf("Description:\n");
        printf("\tProducer Consumer Example.\n");
        printf("Options (one per invocation):\n");
        printf("\t--help\tdisplay this help and exit\n");
        return 0;
    }

    /* Check argument count */
    /* If argument count is greater than 2, then there are too many arguments*/
    if (nargs > 2)
    {
        fprintf(stderr, "%s: too many arguments\n", args[0]);
        return 1;
    }

    /* If argument count is equal to 2, then assign args[1] to count variable */
    if (nargs == 2)
    {
        // Parse through the array of parameters and return 1 if there is a
character other than a number.
        for(i = 0; args[1][i] != '\0'; i++ )
        {
            if (isdigit(args[1][i]) == 0)
            {
                fprintf(stderr, "%s: input parameter should be an
integer.\n", args[0]);
                return 1;
            }
        }

        // Else, it can be safely converted to a number.
        count =  atoi(args[1]);
    }
    //create the process producer and consumer and put them in ready queue.
    //Look at the definitions of function create and resume in exinu/system folder for
reference.

    resume( create(producer, 1024, 20, "producer", 1, count) );
    resume( create(consumer, 1024, 20, "consumer", 1, count) );
}
```

Function Descriptions:

1. resume(create(producer, 1024, 20, "producer", 1, count));
   resume(create(consumer, 1024, 20, "consumer", 1, count));
   Description: The above two lines are called in the main function as displayed above.
   - The create system call is used to create a new process that will execute instructions at the method specified in the first argument. So the first create function will start executing instructions at the producer() method and the second create method will start executing method at the consumer() method.
   - The second parameter in the create system call specifies the stack size i.e. 1024 bytes.
   - The third parameter denotes the priority of the process i.e 20
   - The fourth parameter denotes identifying name i.e. “producer” and “consumer” respectively.
   - The fifth parameter denotes the number of parameters i.e. 1
   - The sixth parameter denotes the actual parameter that is passed to the process. i.e. count
   - This function returns the pid of the created process.
   - The created process is in the suspended state.
   - The resume function accepts the pid of the process and resumes the execution of the process.
   - void producer(int count)
2. void producer(int count)
   - This is the first method passed to the create system call.

    - The producer method accepts one argument count and produces an incremental value starting from 1 to count and assigns it to the extern int variable n.
3. void consumer(int count)
    - This is the second method that is passed to the system call.
    - The consumer method accepts a parameter count then reads and prints the value of n that are assigned by the producer until the value of n = count.

Contributions –

Akshay Kamath (akkamath)

- Worked on question 1 (Does your program output any garbage values)
- Producer source code
- 'MakeFile' changes for including files in app folder
- Debugging the errors and fixing issues. Added error handling for non-numeric characters.

Sameedha Bairagi (sbairagi)

- Worked on question 2 (Are all produced values getting consumed)
- Consumer source code
- Header files
- Debugging the errors and fixing issues