# **2.2 Atividades**

### **1. Estudo e Execução Básica**

* Compile e execute o código com **um produtor e um consumidor** para poucos itens (ex.: -l 10).
* Observe e analise a saída.

**Resposta:** .......................................................................................  
 .......................................................................................  
 .......................................................................................

### **2. Impacto do Buffer**

* Execute o código com:  
  + Buffer de tamanho **1** (-m 1).
  + Buffer de tamanho **10** (-m 10).
* O comportamento se altera? **Explique o porquê.**

**Resposta:** .......................................................................................  
 .......................................................................................  
 .......................................................................................

### **3. Análise de Pausa (Sleep)**

* Adicione uma pausa longa no **consumidor** (ex.: -C 0,0,0,0,0,0,1) e observe o comportamento.

**Perguntas para reflexão:**

* O produtor consegue preencher todo o buffer antes de bloquear?
* Como o **tamanho do buffer** influencia o número de itens que o produtor consegue inserir antes de parar?

**Resposta:** .......................................................................................  
 .......................................................................................  
 .......................................................................................

# **3.2 Atividade**

### **1. Provocando a Falha**

* Crie um cenário que **demonstre a falha do código**.

**Dicas:**

* Use **mais de um consumidor** (-c 2 ou mais).
* Utilize as **flags de pausa** (-C) para orquestrar uma ordem de execução onde **um consumidor “rouba”** o item do buffer antes que o consumidor acordado tenha a chance de executá-lo.

**Tarefa:**

* Descreva a sequência de eventos que leva ao erro (**um consumidor tentando ler de um buffer vazio**).
* Explique por que o uso de while na versão correta previne esse problema.

**Resposta:** .......................................................................................  
 .......................................................................................  
 .......................................................................................

# 

# 

# **4.2 Atividade**

### **1. Provocando o Deadlock**

* Crie um cenário que **cause deadlock**.

**Dicas:**

* Utilize **um produtor** e **dois ou mais consumidores** (-p 1 -c 2).
* Configure as **flags de pausa** (-P e -C) para criar a seguinte sequência:  
  1. Ambos os consumidores bloqueiam esperando por um item.
  2. O produtor insere um item, acorda um consumidor e depois bloqueia porque o buffer (tamanho = 1) está cheio.
  3. O consumidor acordado consome o item e sinaliza… mas acorda o outro consumidor em vez do produtor.

**Tarefa:**

* Explique por que o sistema entra em **deadlock**.
* Justifique como o uso de **duas variáveis de condição (CVs)** na versão correta resolve o problema do **“sinal ambíguo”**.

**Resposta:** .......................................................................................  
 .......................................................................................  
 .......................................................................................