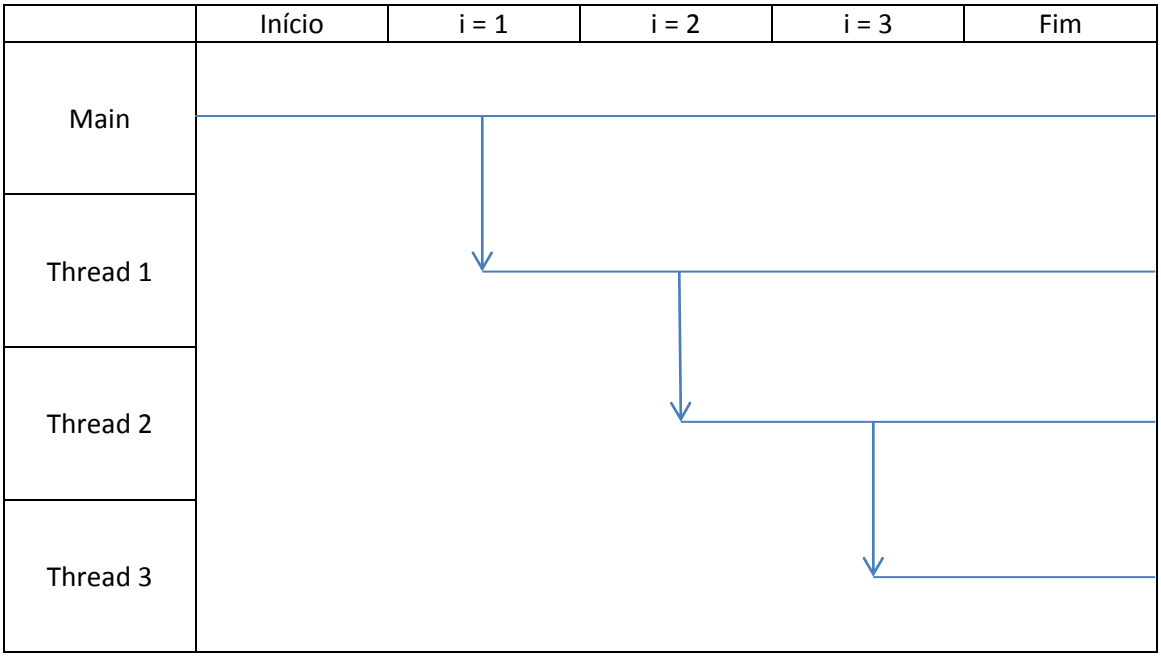
```
/*****************************************************************/
/**  MAC 438  - Programação Concorrente                           **/
/**  IME-USP  - Primeiro Semestre de 2016                         **/
/**  Prof. Marcel Parolin Jackowski                               **/
/**                                                               **/
/**  Primeiro Exercício-Programa                                  **/
/**  Arquivo: Bonus.pdf                                           **/
/**                                                               **/
/**  Ronaldo Yang          7576750                                **/
/**  Yoshio Mori           6432393                                **/
/**                                                               **/
/**  04/04/2016                                                   **/
/*****************************************************************/
```

1)

- Incialmente, vamos supor que todos os processo são criados com sucesso.
- Diagrama:

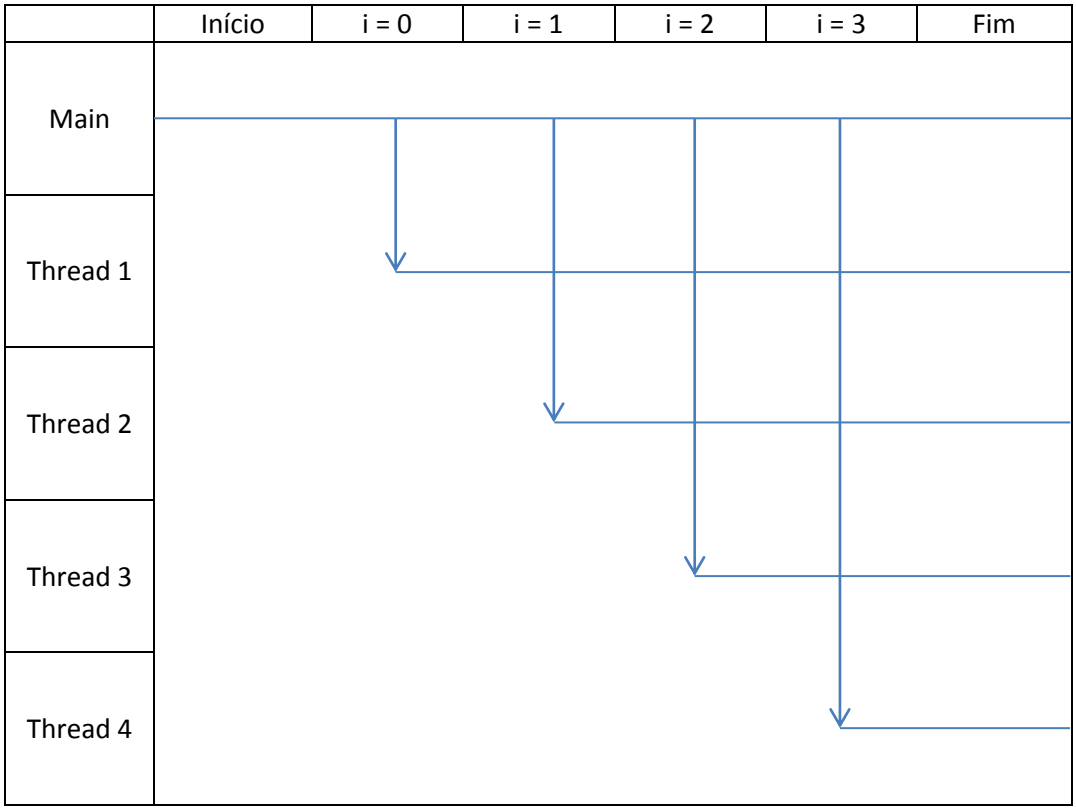


- Inicialmente temos apenas o processo main.
- Quando main entra no loop(i = 1), é criado um processo filho(Thread 1) a partir dele. Assim, como o retorno de fork() não é zero, main sai da repetição e realiza o restante do programa. Já Thread 1, continua realizando o laço.
- Quando Thread 1 entra na próxima iteração do loop(i = 2), é criado um processo filho(Thread 2) a partir dele. E a mesma situação, de main, acontece para Thread 1.
- Quando Thread 2 entra na próxima iteração do loop(i = 3), é criado um processo filho(Thread 3) a partir dele. E a mesma situação, de main e Thread 1, acontece para Thread 2.
- Thread 3 tentara realizar a próxima iteração, mas a mesma terá terminado (i = 4 > n). Assim ela realizará o restante do programa.
- Fim.

- **Observação**: Se fork() falhar, ou seja fork <= -1, as threads seguintes não serão criadas. Por exemplo, se Thread 1 executar fork() e falhar, Thread 2 não será criada e Thread 1 finalizará normalmente o programa. No fim só teremos dois processos: main e Thread 1

2)

- Incialmente, vamos supor que todos os processo são criados com sucesso.
- Diagrama

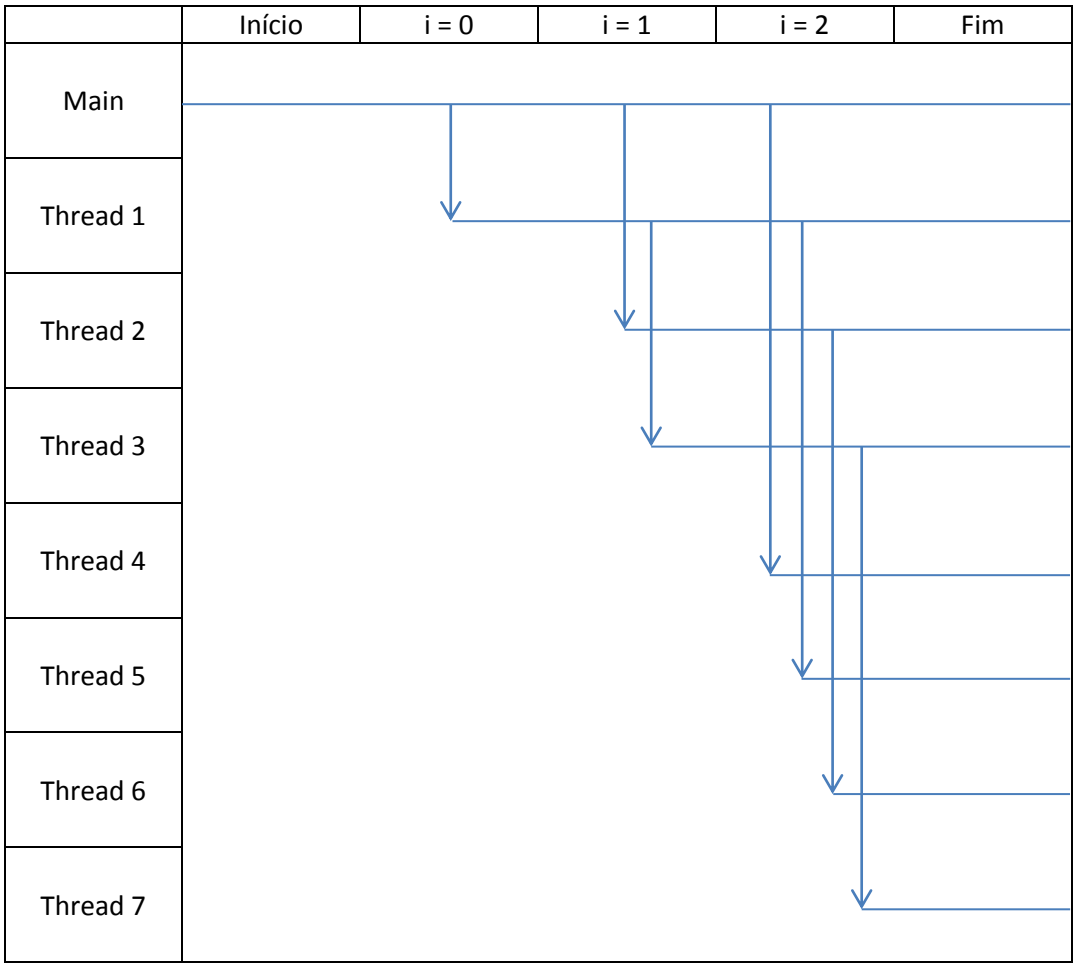


- Inicialmente temos apenas o processo main.
- Quando main entra no loop(i = 0), é criado um processo filho(Thread 1) a partir dele. Para o processo filho(fork() == 0), ele sairá do laço e executará o resto do seu programa. Para o pai, ele continuará iterando.
- O procedimento acima será realizado até i=3. Assim são criados os processos: Thread 2, Thread 3 e Thread 4. Todos filhos de main, assim como Thread 1.
- Fim.

- **Observação**: Se alguma criação de processo falhar, fork() < 0, o processo não será criado, porém main continuará iterando e tentando criar os outros processos filhos.

3)

- Incialmente, vamos supor que todos os processo são criados com sucesso.
- Diagrama:



- Inicialmente temos apenas o processo main.
- Quando main entra no loop(i = 0), é criado um processo filho(Thread 1) a partir dele. Os dois processos, pai e filho, continuam iterando.
- Quando ambos entram na próxima iteração(i=1), ambos criam um processo filho a partir de cada um, Thread 2 para main e Thread 3 para Thread 1. E todos continuam iterando.
- Quando todos entram na próxima iteração(i=2), todos criam um processo filho a partir de cada um, Thread 4 para main, Thread 5 para Thread 1, Thread 6 para Thread 2 e Thread 7 para Thread 3. E todos continuam iterando.
- Como i=3, o laço termina e todos os processos executam o resto do programa.
- No fim, são criados (2^n) – 1 processos.
- Resumindo: todo processo criado cria n-i filhos.

- **Observação**: Se alguma criação de processo falhar, fork() < 0, o processo não será criado e como consequência os seus filhos também não, assim como todos os processos que seriam criados nessa hierarquia. Além disso, a hierarquia de processos que seria criada a partir do processo que realizou a tentativa de criação, será quebrada.