

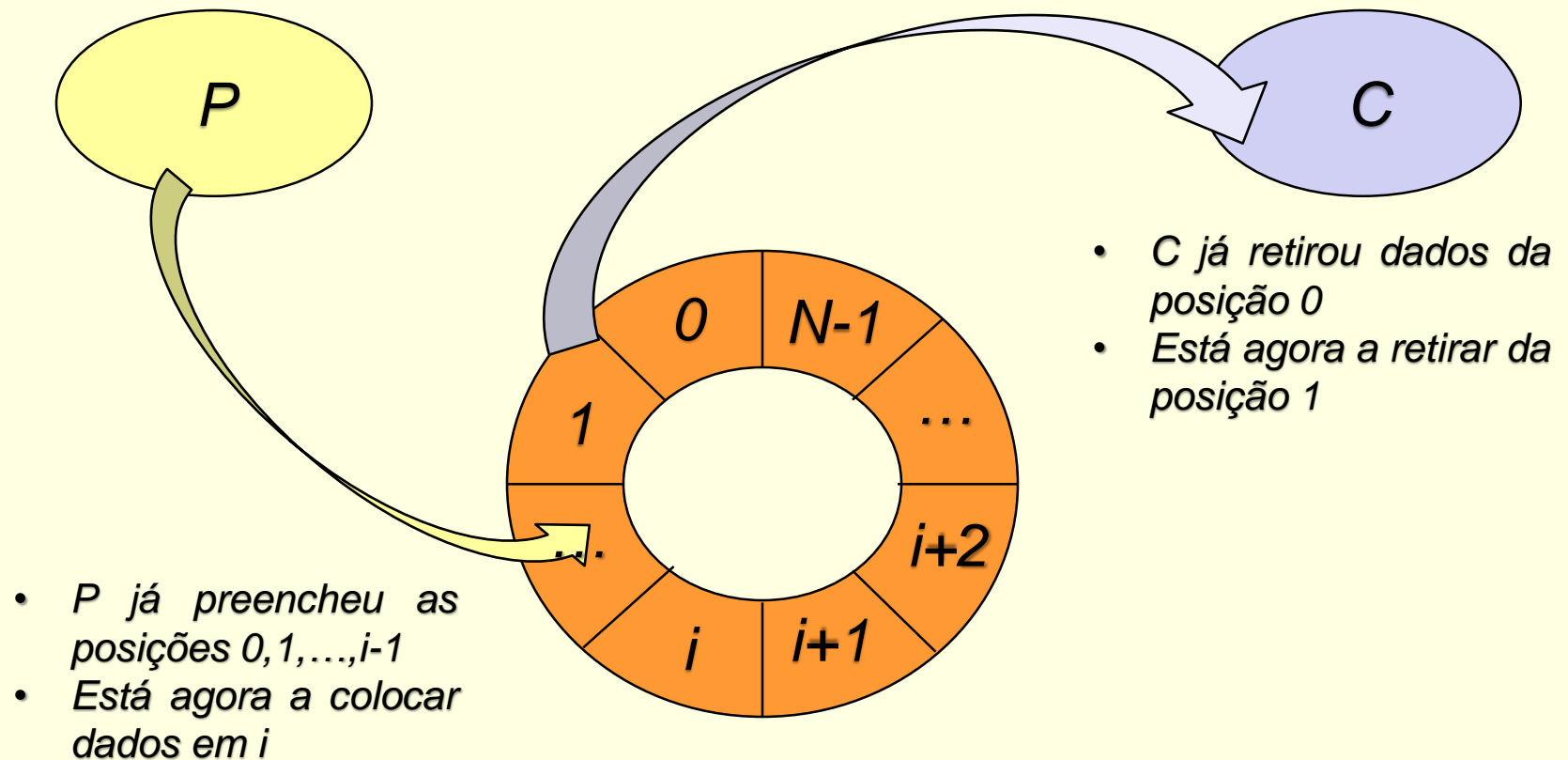
# *Fundamentos de Sistemas de Operação*

---

Unix Windows NT Network MacOS DOS/VS Vax/VMS  
Linux Solaris HP/UX AIX Mach Chorus

*Processos, Threads e o SO:*  
*Produtor/Consumidor com buffer[N]*

# *P/C com buffer* (versão 1)



# *P/C com buffer* (versão 1)

- *p é o índice usado para P meter dados no buffer, i.e.,*
  - $P(p)=\text{dado};$
- *c é o índice usado para C tirar dados do buffer, i.e.,*
  - $\text{dado}=C(c);$
- *Que condições para P e C?*
  - Se cheio, P bloqueia
    - Senão,  $P(p)$  e  $p$  avança - i.e.,  $p = (p++)\%N$
  - Se vazio, C bloqueia
    - Senão,  $C(c)$  e  $c$  avança - i.e.,  $c = (c++)\%N$

# *P/C com buffer* (versão 1)

- *O que são*
  - *cheio? vazio?*
- *Seja*
  - *SlotsOcupados um contador do número de posições ocupadas no buffer e*
  - *SlotsVazios um contador do número de posições vazias no buffer*
- *Então,*
  - *cheio= ( $SlotsOcupados == N$ )*
  - *vazio= ( $SlotsVazios == N$ )*

# *P/C com buffer* (versão 1)

## □ *Produtor*

```

#define CHEIO (slotsCheios==N)

void produz(char c) {
    pthread_mutex_lock(&mtx);
    while CHEIO pthread_cond_wait(&cndP, &mtx);
    putInBuffer(c);
    pthread_cond_signal(&cndC);
    pthread_mutex_unlock(&mtx);
}
    
```

# *P/C com buffer* (versão 1)

## □ Consumidor

```
#define VAZIO (slotsVazios==N)

void consome(char *chr) {
    pthread_mutex_lock(&mtx);
    while VAZIO pthread_cond_wait(&cndC, &mtx);
    getFromBuffer(chr);
    pthread_cond_signal(&cndP);
    pthread_mutex_unlock(&mtx);
}
```

# *P/C com buffer* (versão 1)

## □ Colocar no buffer

```

void putInBuffer(char chr) {
    buffer[p]= chr;
    p= (p+1)%N;
    slotsCheios++; slotsVazios--;
}
    
```

## □ Tirar do buffer

```

void getFromBuffer(char *chr) {
    *chr= buffer[c];
    c= (c+1)%N;
    slotsVazios++; slotsCheios--;
}
    
```

# *P/C com buffer* (versão 1)

## □ *Variáveis (globais)*

```
#define N      32
```

```
char buffer[N];
```

```
int slotsCheios=0, slotsVazios= N;
```

```
int c=0, p= 0;
```

```
pthread_cond_t cndC= PTHREAD_COND_INITIALIZER;
```

```
pthread_cond_t cndP= PTHREAD_COND_INITIALIZER;
```

```
pthread_mutex_t mtx= PTHREAD_MUTEX_INITIALIZER;
```



# *P/C com buffer* (versão 1)

## □ *Threads*

```
void *produtor(void * args) {  
    for (int i= 0; i < 24; i++) produz('A'+i);  
    return NULL;  
}
```

```
void *consumidor(void * args) {  
    char chr;  
    for (int i= 0; i < 24; i++)  
        { consome(&chr); printf("%c ",chr); }  
    return NULL;  
}
```

# *P/C com buffer* (versão 1)

## □ *Principal*

```

int main(int argc, void *argv[]) {
    pthread_t p1, p2;

    pthread_create(&p1, NULL, consumidor, NULL);
    pthread_create(&p2, NULL, produtor, NULL);

    pthread_join(p1, NULL);
    pthread_join(p2, NULL);
    return 0;
}
    
```

# *P/C com buffer* (versão 1)

## □ *Desafio*

- *Tente alterar o código de forma a suportar múltiplos consumidores...*