セマフォを用いたアルゴリズム

1. セマフォを用いた相互排除アルゴリズム

広域変数と初期値

```
semaphore s := 1;        /*  空き資源の数  */
```

プロセス *i* の相互排除処理

```
wait(s);                 /*  資源を要求する  資源が得られない場合は空くまで待つ  */
CSi();                   /*  プロセス i の臨界領域  */
signal (s);              /*  資源を解放する  */
NCSi();                  /*  プロセス i の非臨界領域  */
```

2. セマフォを用いたパイプ処理

広域変数と初期値

```
semaphore s := N;        /*  バッファの空き領域数  */
semaphore m := 0;        /*  バッファ中のメッセージ数  */
Message Buffer[N];       /*  メッセージバッファ  */
```

送信側プロセスの送信処理

```
int i:= 0;                       /*  次に書き込むバッファの位置  */
while (true) {
    send_msg の生成;
    wait (s);                    /*  バッファの空き領域数が 1 以上になるのを待つ  */
    Buffer[i] := send_msg;       /*  バッファへの書き込み  */
    signal (m);                  /*  バッファ内のメッセージ数を 1 増やす  */
    i := (i + 1) mod N;          /*  書き込み位置を 1 つ後ろ(末尾まで行けば先頭)に  */
}
```

受信側プロセスの受信処理

```
int j:= 0;                       /*  次に読み出すバッファの位置  */
while (true) {
    wait (m);                    /*  バッファ内のメッセージ数が 1 以上になるを待つ  */
    recv_msg := Buffer[j];       /*  バッファからの読み出し  */
    signal (s);                  /*  バッファの空き領域数を 1 増やす  */
    j := (j + 1) mod N;          /*  読み出し位置を 1 つ後ろ(末尾まで行けば先頭)に  */
    recv_msg の処理;
}
```

3. セマフォを用いたリーダライタ問題アルゴリズム

広域変数と初期値

```
semaphore w := 1                    /*  書き込みプロセスの制御  */
semaphore m := 1                    /* w, r  に対するアクセス制御  */
int r := 0;                         /*  同時読み出しプロセス数  */
```

ライタプロセスの書き込み処理

```
while (true){
    write_data  の生成;
    wait (w);                       /*  他に読み，書きをしているプロセスがいれば待つ  */
    write (write_data);             /*  書き込み  */
    signal (w);
}
```

リーダプロセスの読み込み処理

```
while (true){
    wait (m);                       /*  他に w,r を操作しているプロセスがいれば待つ  */
    if (r = 0) wait (w);            /*  他のプロセスの書き込みを遮断  */
    ++r;
    signal (m);
    read_data := read();            /*  読み込み  */
    wait (m);
    --r;
    if (r = 0) signal (w);  /*  読み込み中のプロセス数が 0 ならば書き込み可能に  */
    signal (m);
    read_data  の処理;
}
```