相互排除アルゴリズム

1. 厳密な交互実行によるアルゴリズム(2 プロセス)

広域変数と初期値

```
    int turn = 0;                          /* 次に臨界領域に入るプロセス番号 */
```

プロセス 0 の臨界領域処理

```
    while (turn = 1) wait();               /* プロセス 1 が臨界領域から出るまで待つ */
    CS0();                                 /* プロセス 0 の臨界領域 */
    turn := 1;                             /* 次はプロセス 1 が臨界領域に入る番 */
    NCS0();                                /* プロセス 0 の非臨界領域 */
```

プロセス 1 の臨界領域処理

```
    while (turn = 0) wait();               /* プロセス 0 が臨界領域から出るまで待つ */
    CS1();                                 /* プロセス 1 の臨界領域 */
    turn := 0;                             /* 次はプロセス 0 が臨界領域に入る番 */
    NCS1();                                /* プロセス 1 の非臨界領域 */
```

2. Dekker の相互排除アルゴリズム(2 プロセス)

広域変数と初期値

```
    int turn := 0                          /* 次に臨界領域に入るプロセス番号 */
    boolean flag0 := false, flag1 := false;  /* 臨界領域に入ることの宣言 */
```

プロセス 0 の臨界領域処理

```
    flag0 := true;                         /* 臨界領域に入ると宣言 */
    while (flag1) {                        /* プロセス 1 が臨界領域に入っているか? */
        if (turn = 1) {                    /* プロセス 1 の番か? */
            flag0 := false;                /* 宣言を取り下げ */
            while (turn = 1) wait();       /* プロセス 1 が臨界領域を出るまで待つ */
            flag0 := true;                 /* 再度宣言 */
        }
    }
    CS0();                                 /* プロセス 0 の臨界領域 */
    turn := 1;                             /* 次はプロセス 1 が臨界領域に入る番 */
    flag0 := false;
    NCS0();                                /* プロセス 0 の非臨界領域 */
```

プロセス 1 の臨界領域処理

(省略)

3. Peterson の相互排除アルゴリズム(2 プロセス)

広域変数と初期値

```
int turn := 0                                   /*  次に臨界領域に入るプロセス番号  */
boolean flag0 := false, flag1 := false;         /*  臨界領域に入ることの宣言  */
```

プロセス 0 の臨界領域処理

```
flag0 := true;                                  /*  臨界領域に入ると宣言  */
turn := 1;                                      /* P1 の番にする  */
while (flag1 and (turn = 1)) wait();            /* P1 の番かつ P1 も臨界領域なら待つ  */
CS0();                                          /* P0 の臨界領域  */
flag0 := false;
NCS0();                                         /* P0 の非臨界領域  */
```

プロセス 1 の臨界領域処理

(省略)

4. Lamport のパン屋のアルゴリズム(*N* プロセス)

広域変数と初期値

```
int priority[N] := {0, 0, ..., 0};              /*  優先順位(0 以外で最小の値が優先) */
boolean enter[N] := {false, false, ...,false};  /*  優先順位取得中  */
```

プロセス *i* の臨界領域処理

```
enter[i] := true;                               /*  優先順位取得開始  */
priority [i] := 1 + max {priority [0], priority [1], ..., priority [N-1]};
            /* (それまでに他のプロセスが得た優先順位)+1 を優先順位として得る  */
enter[i] := false;                              /*  優先順位取得完了  */
for (int j := 0; j < N; ++j) {
    while (enter[j]) wait();                    /*  プロセス j が順位を得るまで待つ  */
    while ((priority [j]  ≠  0) and ((priority [j], j) < (priority [i], i))) wait();      (※)
                    /*  自分より優先順位の高いプロセスが処理を終えるのを待つ  */
}
CSi();                                          /*  プロセス i の臨界領域  */
priority [i] := 0;                              /*  優先順位をリセット  */
NCSi();                                         /*  プロセス i の非臨界領域  */
```

(※) (*priority* [*j*], *j*) < (*priority* [*i*], *i*) は

(*priority* [*j*] < *priority* [*i*]) or (*priority* [*j*] = *priority* [*i*] and *j* < *i*) のこと

(まず優先順位を比較、優先順位が同じならプロセス番号を比較)