オペレーティングシステム H25 年度 後期末試験

(2014.02.06 重村 哲至)

IE4 ____ 番 氏名 ____模範解答____

1. 次の文の空欄に最も適切な語句を語群より記号で選びなさい。(2 点× 25 問=50 点)

複数のプロセスが情報交換しながら一つの仕事を成し遂げる場合は、プロセスの (1) が必要になる。一方、複数のプロセスが関係のない処理をしている場合でも、知らないうちに互いに影響をあたえる場合がある。このような場合は、プロセスの (2) が生じている。

複数のプロセスが共通の資源を取り合う場合は、プロセスの (3) が生じる。これにより誤った処理結果にならないようにするためには、プログラム中で共通資源にアクセスする部分である (4) が同時に実行されないように (5) をする必要がある。

(5) のためには Dekker のアルゴリズム等を用いることも可能であるが、ユニプロセッサ (シングルプロセッサ) のシステムでは (4) を (6) 状態にして実行する方法が実用的である。マルチプロセッサのシステムでは (7) 命令や (8) 命令のような、特別な (9) 命令を用いる。

Dekker のアルゴリズム等は (10) を用いる原始的な (5) 機構である。それに対しセマフォはプロセスの (11) を伴う、より高度な (5) 機構である。セマフォに対しては、Wait または lock の役割を持つ (12) 命令と、Signal または unlock の役割を持つ (13) 命令が定義される。(12) 命令は、セマフォの値が (14) のとき値を 1 減じる。セマフォの値が 0 のときは、プロセスをセマフォの (15) に入れ実行をブロックする。

メッセージ転送機構は、ユーザプロセスに使いやすい (16) 間通信を提供する。通信相手の指定方法には、プロセスの名前を用いる (17) 指定方式と、通信チャネルの名前を用いる (18) 指定方式がある。また、メッセージの送受信時に (16) がブロックすることがある (19) 方式と、ブロックすることがない (20) 方式に分類される。

複数の資源を同時に使用する場合、資源を 1 つずつ順に確保すると (21) が発生することがある。これは、プロセスが一部の資源を確保したままで別の資源の待ちになる (22) が原因である。(22) は資源を確保する (23) を全プロセスで統一することで解決できるが、(22) の中でも特殊な (24) は解決が難しい。解決には一度に複数の資源を確保できる (25) 命令を導入するなど、機構を拡張することが必要である。

語群：(あ) 同期、(い) 非同期、(う) 直接、(え) 間接、(お) 競合、(か) 協調、(き) 干渉、(く) 順序、(け) 機械語、(こ) 排他制御、(さ) 状態遷移、(し) 割込み許可、(す) 割込み禁止、(せ) 確保待ち、(そ) 待ち行列、(た) プロセス、(ち) クリティカルセクション、(つ) デッドロック、(て) ビジーウェイティング、(と)Swap、(な)Test And Set、(に)Load Effective Address、(ぬ)P、(ね)P_and、(の)P_or、(は)V、(ひ)V_and、(ふ)V_or、(へ)0 未満、(ほ)1 未満、(ま)1 以上、(み) 循環待ち

(1)	か	(2)	き	(3)	お	(4)	ち	(5)	こ	(6)	す	(7)	と、な
(8)	な、と	(9)	け	(10)	て	(11)	さ	(12)	ぬ	(13)	は	(14)	ま
(15)	そ	(16)	た	(17)	う	(18)	え	(19)	あ	(20)	い	(21)	つ
(22)	せ	(23)	く	(24)	み	(25)	ね						

2. 次のプロセスが並行実行されたときの出力を答えなさい。なお、両方のプロセスの出力は、出力された順に同じ画面に表示されるものとします。(完全にできて、10 点)

```
      SEMAPHORE S = 0; // 初期値 0 のセマフォ
      SEMAPHORE M = 1; // 初期値 1 のセマフォ

Process A             |  Process B
  P(S);               |    P(M);
  printf("A-1\n");    |    printf("B-1\n");
  V(M);               |    V(S);
  P(S);               |    P(M);
  printf("A-2\n");    |    printf("B-2\n");
  V(M);               |    V(S);
```

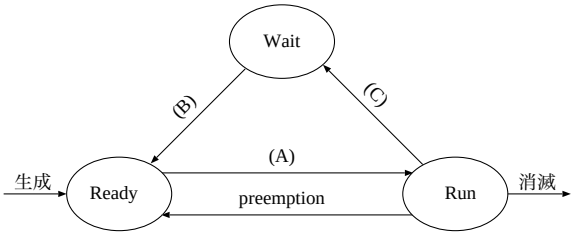
B-1

A-1

B-2

A-2

3. プロセスの状態遷移図に関する問題に答えなさい。
(2 点 × 4 問 = 8 点)



以下の遷移が図中のどれに当たるか、「(A)」、「(B)」、「(C)」、「該当なし」のいずれかで答えなさい。

(1) 値 1 のセマフォに P 命令を実行したために、自プロセスに起こる遷移

該当なし

(2) 値 0 のセマフォに P 命令を実行したために、自プロセスに起こる遷移

(C)

(3) 値 1 のセマフォに V 命令を実行したために、他プロセスに起こる可能性のある遷移

該当なし

(4) 値 0 のセマフォに V 命令を実行したために、他プロセスに起こる可能性のある遷移

(B)

4. 次の C 言語風のプログラムは、単一プロデューサ/単一コンシューマ問題のセマフォを用いた解を示しています。空欄に補う P 命令か V 命令の記述を答えなさい。(8 点)

```
SEMAPHORE S = 5; // セマフォS:初期値 (5)
SEMAPHORE M = 0; // セマフォM:初期値 (0)
MESSAGE BUF[5];  // 大きさ 5 のバッファ

producer() {
  int i = 0;
  while(true) {
    /* (A)  */
    BUF[i]=メッセージ;
    /* (B)  */
    i = (i + 1) % 5;
  }
}

consumer() {
  int j = 0;
  while(true) {
    /* (C) */
    メッセージ=BUF[j];
    /* (D) */
    j = (j + 1) % 5;
  }
}
```

(A)	P(S);
(B)	V(M);
(C)	P(M);
(D)	V(S);

5. 次の C 言語風のプログラムは、複数プロデューサ/複数コンシューマ問題のセマフォを用いた解を示しています。空欄に補う P 命令か V 命令の記述を答えなさい。(8 点)

```
SEMAPHORE S = 5; // セマフォS:初期値 (5)
SEMAPHORE M = 0; // セマフォM:初期値 (0)
MESSAGE BUF[5];  // 大きさ 5 のバッファ

SEMAPHORE A = 1; // セマフォA:初期値 (1)
int i = 0;
producer() {
  while(true) {
    /* (A)  */
    /* (B)  */
    BUF[i]=メッセージ;
    /* (C)  */
    i = (i + 1) % 5;
    /* (D)  */
  }
}

SEMAPHORE B = 1; // セマフォB:初期値 (1)
int j = 0;
consumer() {
  while(true) {
    /* (E) */
    /* (F) */
    メッセージ=BUF[j];
    /* (G) */
    j = (j + 1) % 5;
    /* (H) */
  }
}
```

(A)	P(A);	(B)	P(S);
(C)	V(M);	(B)	V(A);
(E)	P(B);	(B)	P(M);
(G)	V(S);	(B)	V(B);

6. 次の C 言語風のプログラムは、複数リーダ/複数ライタ問題のセマフォを用いたリーダ弱優先解を示しています。空欄に補う P 命令か V 命令の記述を答えなさい。(8 点)

```
SEMAPHORE W = 1; // セマフォW:初期値 (1)
DATA BUF;        // 共有データ

writer() {
  int i = 0;
  while(true) {
    /* (A) */
    BUF = データ;
    /* (B) */
  }
}

// リーダの共通変数
SEMAPHORE M = 1; // セマフォM:初期値 (1)
int r = 0;

reader() {
  while(true) {
    /* (C) */
    if (r==0) /* (D) */
    r = r + 1;
    /* (E) */
    データ = BUF;
    /* (F) */
    r = r - 1;
    if (r==0) /* (G) */
    /* (H) */
  }
}
```

(A)	P(W);	(B)	V(W);
(C)	P(M);	(B)	P(W);
(E)	V(M);	(B)	P(M);
(G)	V(W);	(B)	V(M);

7. 次の C 言語風のプログラムは、複数プロデューサ/複数コンシューマ問題のシーケンサとイベントカウントを用いた解を示しています。ticket(S) は、シーケンサ S の値を取り出し、その後、シーケンサの値を 1 進めます。await(E,v) は、イベントカウント E の値が v になるまでプロセスを待ち状態にします。advance(E) は、イベントカウントの値を進めて待っていたプロセスを起こします。空欄に補う記述を答えなさい。(8 点)

```
SEQUENCER  T = 0;   // シーケンサ T:初期値 (0)
SEQUENCER  U = 0;    // シーケンサ U:初期値 (0)
EVENTCOUNT IN = 0;
EVENTCOUNT OUT = 0;
MESSAGE BUF[5];      // 大きさ 5 のバッファ

producer() {
  int t;
  while(true) {
    // 他のプロデューサと同期をとる
    t = ticket(T);
    await(IN, t);
    // コンシューマと同期をとる
    await(OUT, /* (A) */ );
    BUF[t % 5]=メッセージ;
    advance( /* (B) */ );
  }
}

consumer() {
  int u;
  while(true) {
    // 他のコンシューマと同期をとる
    u = ticket(U);
    await(OUT, u);
    // プロデューサと同期をとる
    await(IN, /* (C) */ );
    メッセージ=BUF[u % 5];
    advance( /* (D) */ );
  }
}
```

(A)	t-4
(B)	IN
(C)	u+1
(D)	OUT