

トロン技術者認定試験問題



00030122

A. 注意事項

1. **試験開始の合図があるまで、この問題冊子を開いてはいけません。**
2. 試験開始と終了の時刻は試験監督員の時計が基準となります。試験監督員の指示に従いなさい。
3. 本冊子の本文は23ページです。試験中にページの落丁、乱丁、印刷不鮮明、および解答用紙の汚れ等に気づいた場合は手を挙げて試験監督員に知らせなさい。
4. **問題に関する質問には答えることはできません。** 文意どおりに解釈して解答しなさい。
5. 「受験票」は、必ず机の上に置きなさい。
6. 「受験票」のほかに試験時間中、机の上に置けるものは、「B または HB の黒鉛筆または黒色シャープペンシル」「プラスチック製の消しゴム」「鉛筆削り（電動式を除く）」「時計」「眼鏡」です。これ以外の所持品を置いてはいけません。
7. 解答用紙には氏名欄があるので、試験監督員の指示に従って、正しく記入しなさい。
8. 解答用紙の試験会場欄、受験日欄が正しいことを、試験監督員の指示に従って確認しなさい。
9. 解答用紙の受験番号欄に、自分の受験番号が記載されており、かつ、正しくマークされていることを確認しなさい。
10. 13時30分以降は、試験終了後の確認作業が終了するまで退室を認めません。**試験中の発病又はトイレ等やむを得ない場合には、手を挙げて試験監督員の指示に従いなさい。**ただし、一時退室が認められた場合でも、試験教室以外での受験はできません。試験時間の延長も認められません。
11. 試験時間中は、試験監督員の指示に従いなさい。従わない場合は退室させることがあります。

B. 本冊子、解答用紙について

1. 本冊子は、トロン技術者認定試験のためのものです。
2. 本冊子には、計25問の問題が収録されています。**すべての問題を解答しなさい。**
3. 解答用紙は本冊子とは別に1枚あり、マークシート方式による記入となります。
4. 本冊子の余白等は適宜利用してよいですが、どのページも切り離してはいけません。
5. **本冊子、解答用紙は持ち帰ることはできません。**
6. 以下の欄に受験番号と氏名を記入しなさい。

受験番号	
氏名	

解答上の注意が裏表紙に記載してあるので、この問題冊子を**左側から裏返**して必ず読みなさい。ただし、問題冊子を開いてはいけません。

注意事項

- ・ 特に記載されていない限り、 μ ITRON とは μ ITRON4.0 と考えなさい。
- ・ 特に記載されていない限り、「タスク」とはプログラムの並行実行の単位と考えなさい。
- ・ 特に記載されていない限り、「システムコール」と「サービスコール」は同じ意味と考えなさい。
- ・ 特に記載されていない限り、マルチプロセッサでの動作は考慮せずに解答しなさい。
- ・ プログラムリストに含まれる /* 略 */ の箇所は、何らかの処理が省略されているものとして解答しなさい。

出題区分 1 (20問 各3点)

問 1

T-Kernel (μ ITRON) のタスク管理機能に関する以下の説明のうち、誤っているものを一つ選べ。
※ μ ITRON の場合は () 内のサービスコールを呼び出すものとして解答せよ。

選択肢

1. タスク管理機能は、タスクの状態を直接操作したり、参照したりする機能である。タスクが生成される前、あるいは削除された後のタスクの状態を未登録状態という。タスク管理機能のひとつであるタスク生成とは、未登録状態のタスクを休止状態へ移行することである。
tk_sta_tsk (sta_tsk) は休止状態のタスクを、実行可能状態へ移行する。
2. タスク例外処理機能は、タスク実行中に例外的な事象が発生した場合に、タスクの一時的な処理をするルーチンを起動できる機能である。タスク例外処理はタスク例外が発生したタスクと同じコンテキストで実行されるので、タスク例外処理の中で自タスクを終了させることもできる。
3. tk_chg_pri (chg_pri) は、他タスクのタスク優先度を変更する。自タスクの優先度を変更することはできない。自タスクの優先度を変更する場合は、他タスクから、tk_chg_pri (chg_pri) を自タスクに発行してもらう必要がある。これは、tk_chg_pri (chg_pri) 発行により、自分自身の実行権を失うことを回避するためである。
4. tk_ext_tsk (ext_tsk) は、自タスクを正常終了させ、休止状態へと移行させる。tk_ext_tsk (ext_tsk) は自タスクの終了と削除を行う。tk_ext_tsk (ext_tsk) および tk_ext_tsk (ext_tsk) で終了した場合、タスクが獲得した資源は自動的に解放されない。

問 2

T-Kernel (μ ITRON)のイベントフラグに関する以下の説明のうち、誤っているものを一つ選べ。

選択肢

1. イベントフラグは、イベントの有無をビットごとのフラグで表現することで同期を行うための機能である。イベントフラグは ID 番号で識別されるオブジェクトである。
2. イベントフラグの待ち行列は、FIFO 順、またはタスクの優先度順を、イベントフラグを待つ時に設定することができる。
3. イベントフラグでは、一つのイベントフラグで同時に複数のタスクが待ち状態になることができる。しかし、オプションの指定で、同時に複数のタスクが待ち状態になることを許さない設定もできる。
4. イベントフラグ待ち状態のタスクは、待ち条件を満たせば、先頭でなくても待ち状態が解除されることがある。このため、待ち行列につながれた順番で待ちが解除されるとは限らない。

問 3

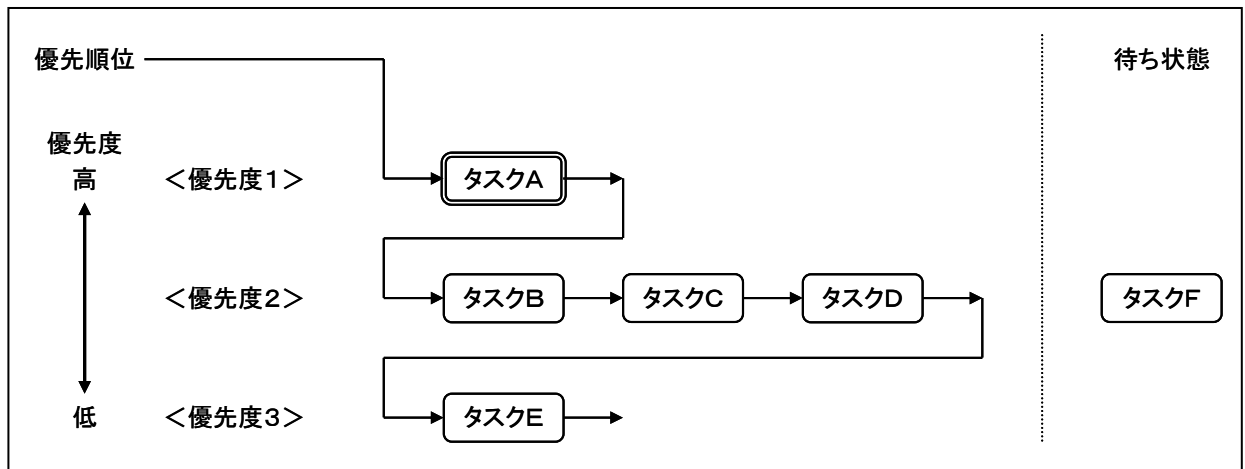
T-Kernel (μ ITRON)のメモリプール管理機能に関する以下の説明のうち、誤っているものを一つ選べ。

選択肢

1. 使用していないメモリを蓄えておき、必要になった場合に提供するメモリを「メモリブロック」と呼ぶ。T-Kernel や μ ITRON のメモリプール機能は、ソフトウェアによって動的なメモリ管理を行う機能である。固定長メモリプール、可変長メモリプールの 2 種類の機能がある。
2. 固定長メモリプールは、固定されたサイズのメモリブロックを動的に管理するためのオブジェクトである。固定長メモリプールとして利用するメモリ領域と、メモリブロックの獲得を待つタスクの待ち行列を持つ。
3. 可変長メモリプールを利用すると、小さな空き領域がたくさん残ってしまうメモリリークという問題が発生する可能性がある。これは、小さなメモリブロックをたくさん開放して空き領域を作っても、連続した大きな空き領域が存在しないため、大きなメモリブロック要求に対応できず、大きなメモリブロックを要求したタスクが待ち状態からぬけだせないというものである。
4. 固定長メモリブロックの獲得要求をするサービスコールでは、ポーリングによる処理やタイムアウトを指定することができる。

問 4

以下の図は、T-Kernel や μ ITRON を利用したシステムにおけるタスクの優先度と優先順位の関係を表す例である。



図では具体的に以下の内容を表わしている。

- ・ 優先順位はタスク A、タスク B、タスク C、タスク D、タスク E の順になっている。
- ・ タスク A は実行状態、タスク B、タスク C、タスク D、タスク E は実行可能状態になっている。
- ・ タスク F は待ち状態になっている。
- ・ タスク A の優先度は 1 (高優先度) である。
- ・ タスク B、タスク C、タスク D、タスク F の優先度は 2 (中優先度) である。
- ・ タスク E の優先度は 3 (低優先度) である。

この状態において、タスク F の待ち状態が解除された場合の動作に関する説明として、ふさわしいものを選択肢の中から一つ選べ。

なお、ここで言う優先度とは「タスク優先度」を表わしている。タスク優先度は、タスクの処理順序を制御するために与えられるパラメータである。一方、優先順位とは、処理の実行順序を明確にするための概念であり、タスクの優先度に基づいて定められる。

選択肢

1. タスク F は実行状態となる。タスク F の優先順位はタスク A の前に挿入される。タスク A は待ち状態に移行する。タスク B～タスク E の状態は変化しない。タスク A～タスク E の優先度は変化しない。タスク F の優先度は 2 から 1 に変化する。
2. タスク F は実行可能状態となる。タスク F の優先順位はタスク A とタスク B の間に挿入される。タスク A～タスク E の状態は変化しない。タスク A～タスク E の優先度は変化しない。タスク F の優先度は 2 から 1 に変化する。
3. タスク F は実行可能状態となる。優先順位はタスク A とタスク B の間になる。タスク A～タスク E の状態は変化しない。すべてのタスクの優先度は変化しない。
4. タスク F は実行可能状態となる。優先順位はタスク D とタスク E の間になる。タスク A～タスク E の状態は変化しない。すべてのタスクの優先度は変化しない。

問 5

異常処理用の通信データを通常の通信データよりも先に処理したい場合など、メッセージに優先度を設定できると便利なケースがある。

T-Kernel や μ ITRON において、このような用途にふさわしいオブジェクトを以下の中から一つ選べ。

選択肢

1. メールボックス
2. イベントフラグ
3. メッセージバッファ
4. ミューテックス

問 6

誤り検出や誤り訂正に関する説明として、誤っているものを一つ選べ。

選択肢

1. 1ビットの水平パリティと1ビットの垂直パリティを組み合わせることで1ビットの誤りを訂正することができる。
2. ハミング符号を使った誤り制御は、誤りを検出することはできるが、訂正することはできない。
3. CRC方式による誤り制御では、誤りを訂正することはできない。
4. CRC方式による誤り制御によってバースト誤りを検出することができる。

問 7

T-Kernel (μ ITRON)の同期・通信機能に関する以下の説明のうち、正しいものをすべて選べ。

選択肢

1. セマフォの初期値を0として、セマフォを生成することができる。
2. イベントフラグの属性に TA_WSGL を指定した場合、複数のタスクが同時に待ち状態になることが禁止される。
3. メールボックスで送受信されるメッセージの本体は、送受信時に内容がコピーされる。
4. ミューテックスは、優先度継承プロトコルと優先度上限プロトコルをサポートする。

問 8

スイッチのチャタリングに関する以下の説明のうち、誤っているものをすべて選べ。

選択肢

1. チャタリングとは、機械的なスイッチの ON/OFF を操作する際に、接点がバウンドするなどの機械的な振動により、短い時間間隔で複数回の ON/OFF を繰り返す現象である。
2. デジタルカメラのシャッターボタンのように、リアルタイムな反応を要求されるスイッチの場合、チャタリングによる誤動作を除去することは反応速度の低下を招くため、適切ではない。
3. チャタリングによる誤動作をソフトウェア的に除去する方法はいろいろ考えられるが、いずれも時間的な待ちを入れることによって、余分な ON/OFF を取り除くものである。
4. スwitchの ON/OFF の信号を直接割込み入力信号とした場合、スイッチのチャタリングにより割込みハンドラが短時間に複数回起動されてしまう。このため、スイッチからの入力を割込み入力信号とする場合には、チャタリングを除去するハードウェア回路を追加しなければならない。

問 9

T-Kernel の時間管理機能に関する以下の説明のうち、誤っているものを一つ選べ。

選択肢

1. システム動作中に `tk_set_tim` を使ってシステム時刻を更新した場合、タイムアウトのパラメータで指定された相対時間は変化しない。たとえば、60 秒後にタイムアウトするように指定した場合、タイムアウト待ちの間に `tk_set_tim` で時間を 60 秒進めてもそこでタイムアウトすることはなく、タイムアウトの指定から 60 秒後にタイムアウトする。
2. `tk_get_tim` ではシステム時刻の現在の値を読み出す。システム時刻は、1985 年 1 月 1 日 0 時 0 分 0 秒 (GMT) からの通算の秒数とする。
3. `tk_get_otm` はシステム稼働時間を取得する。システム稼働時間は、システム起動時からの単純増加する時間であり、`tk_set_tim` による時刻設定に影響されない。
4. `tk_set_tim`、`tk_get_tim`、および `tk_get_otm` システムコールでは、時刻を指定する。時刻の指定には以下のように定義されたシステム時刻型 (SYSTIM 型) が用いられる。

```
typedef struct systim {
    W      hi;
    UW     lo;
} SYSTIM;
```

問 10

以下は、 μ ITRON を利用して書かれたプログラムの一部である。

このプログラムを実行した場合、TaskA~TaskD の起動順はどのようになるか。正しいものを選択肢の中から一つ選べ。

ただし、プログラムは以下の条件で動作するものとする。

- ・ TaskIni から実行が開始される。
- ・ 各タスクの初期状態、初期優先度は以下の通りとする。

タスク	初期状態	初期優先度
TaskIni	実行状態	1
TaskA	休止状態	2
TaskB	休止状態	2
TaskC	休止状態	2
TaskD	休止状態	2

※ T-Kernel の場合は、以下を前提として解答せよ。

- ・ システムコールの名称は `sta_tsk` を `tk_sta_tsk` のように読み替える。

```
void TaskIni( VP_INT exinf )
{
    sta_tsk(TASK_A_ID, 0); /* TaskA を起動 */
    sta_tsk(TASK_B_ID, 0); /* TaskB を起動 */
    sta_tsk(TASK_C_ID, 0); /* TaskC を起動 */
    sta_tsk(TASK_D_ID, 0); /* TaskD を起動 */
    rot_rdq(2);
    ext_tsk();
}
```

選択肢

1. TaskA → TaskB → TaskC → TaskD
2. TaskB → TaskC → TaskD → TaskA
3. TaskC → TaskD → TaskA → TaskB
4. TaskD → TaskC → TaskB → TaskA

問 1 1

以下はフローコントロールに関する説明文である。

文中の空欄 [ア] ~ [ウ] を埋めて説明を完成させる場合の正しい組合せを、選択肢の中から一つ選べ。

データ転送におけるフローコントロールは、データの取りこぼしなどを防止する仕組みの一つであり、一般に [ア] 制御と呼ばれる方法と [イ] 制御と呼ばれる方法がある。

受信側の能力が例えば他の割り込み処理などで一時的に低下し、受信能力を超えるまたは超えそうになった場合に、受信側から送信側に通知し、送信側はデータの再送または送信の一時停止などの対応を行う。

再送を行う方式を Ack/NAck 制御と呼び、受信側は正常に受信処理できた場合は Ack を、異常時には NAck を返答として送信側に通知する。送信側は NAck を受けた場合には、Ack 応答があるまで対応するデータを再送することで、データの転送を完了させる。

別の方式に Ready 制御方式がある。この方式では受信側はあらかじめ受信能力が十分に確保できる場合に Ready を、それ以外の場合は Not Ready を送信側に通知する。Not Ready を検出した場合、送信側はデータの送信を停止する。Ready と逆の論理である Busy を使用する場合もあるが、論理が異なるのみで動作原理は同じである。

Ready 制御では常に Ready 状態を監視する必要がある。通信経路が長いなど Not Ready の伝達が遅くなる場合には [ウ] が必要なデータを保持しておかねばならない。

一方、Ack/NAck 方式は状態判断を伴わない送信が可能である。しかし、通信経路が長いなど Ack 応答までの時間が長い場合には [エ] が必要なデータを保持しておかねばならない。

選択肢

	[ア]	[イ]	[ウ]	[エ]
1.	開ループフロー	接続許可	送信側	受信側
2.	輻輳	再送	送信側	受信側
3.	ハードウェアフロー	ソフトウェアフロー	受信側	送信側
4.	優先度フロー	帯域	受信側	送信側

問 1 2

T-Kernel (μ ITRON)の時間管理機能に関する以下の説明のうち、誤っているものを一つ選べ。

※ μ ITRONの場合は()内のサービスコールを呼び出すものとして解答せよ。

選択肢

1. アラームハンドラは、指定した時刻に起動されるタイムイベントハンドラである。アラームハンドラの生成時は、アラームハンドラの起動時刻は設定されておらず動作は停止している。アラームハンドラの起動時刻は、tk_sta_alm (sta_alm) が呼び出された時刻からの指定された相対時刻後として設定される。
2. 動作状態になっているアラームハンドラに対して tk_sta_alm (sta_alm) を発行した場合、既に設定されていた起動時刻を解除し、新たに指定された起動時刻が設定される。
3. 周期ハンドラは、一定周期で起動されるタイムイベントハンドラである。周期ハンドラは生成時の属性に TA_STA を指定することで、周期ハンドラを生成した時から動作している状態とすることができる。
4. 周期ハンドラが動作していない状態において周期ハンドラが起動する時刻になると、周期ハンドラは動作を開始し、次に起動すべき時刻の決定を行う。

問 1 3

T-Kernel (μ ITRON)のタスク管理機能に関する以下の説明のうち、正しいものをすべて選べ。

※ μ ITRONの場合は()内のサービスコールを呼び出すものとして解答せよ。

選択肢

1. T-Kernel (μ ITRON)仕様のスケジューリング規則では、優先順位の高いタスクが実行できる状態にある限り、それより優先順位の低いタスクが実行されることはない。
2. メモリの獲得・解放を行うシステムコールを用いて、メモリの獲得・解放などを行っている状態のときに、タスクの強制終了を行った場合においても、T-Kernel (μ ITRON)の仕様上は動作が保障されている。
3. 休止状態のタスクを指定して tk_ter_tsk (ter_tsk) を発行した場合、対象となるタスクのタスク状態は休止状態のまま変化せず、tk_ter_tsk (ter_tsk) には E_OK が返される。
4. T-Kernel (μ ITRON)では、WAITING、WAITING-SUSPENDED、SUSPENDED を区別している。この理由は、WAITING、SUSPENDED の両方が重なって起こる場合があるからであり、WAITING-SUSPENDED は、両方の状態が起こっていることを明示的に示すために用意されている。

問 1 4

T-Kernel (μ ITRON)において、自タスクを終了させる方法に関する以下の説明のうち、正しいものをすべて選べ。

※ μ ITRON の場合は () 内のサービスコールを呼び出すものとして解答せよ。

※ μ ITRON では起動要求キューイング数は0とする。

選択肢

1. tk_ext_tsk (ext_tsk) を呼び出すことによって自タスクを終了する。
2. tk_exd_tsk (exd_tsk) を呼び出すことにより自タスクを終了し、さらに自タスクを削除する。
3. 引数として自タスクを指定し tk_ter_tsk (ter_tsk) を呼び出すことにより自タスクを終了する。
4. 自タスクのタスク例外処理ルーチンで tk_del_tsk (del_tsk) を呼び出すことによって自タスクを削除する。

問 1 5

以下は T-Kernel のサブシステムとデバイスドライバのいずれかについて、その特徴的な点を説明したものである。

この中で、デバイスドライバに関する説明は以下のいずれか。正しいものを一つ選べ。

選択肢

1. アプリケーションからこの機能呼び出す際の API が定められている。
2. プロセス毎のリソースを管理する機能がある。
3. タスク例外の発生時に呼び出されるブレーク関数を定義することができる。
4. T-Kernel/SM の機能ではなく、T-Kernel/OS の機能に含まれる。

問 1 6

T-Kernel (μ ITRON)のタスク状態に関する以下の説明のうち、誤っているものを一つ選べ。
※ μ ITRONの場合は()内のサービスコールを呼び出すものとして解答せよ。

選択肢

1. T-Kernel (μ ITRON)のタスク状態は、大きく、実行状態、実行可能状態、広義の待ち状態、休止状態、未登録状態の5つに分類される。
2. 待ち状態のタスクに対して、tk_sus_tsk (sus_tsk) を発行し、その後 tk_rsm_tsk (rsm_tsk) を発行したときのタスク状態は実行可能状態になる。
3. 二重待ち状態のタスクに対して、待ち状態が解除された場合、そのタスク状態は強制待ち状態になる。
4. 強制待ち状態のタスクに対して、tk_ter_tsk (ter_tsk) を発行した場合、そのタスク状態は休止状態になる。

問 1 7

T-Kernel Standard Extension のプロセス/タスク管理機能に関する以下の説明のうち、正しいものをすべて選べ。

選択肢

1. プロセスの状態として未生成状態、実行可能状態、実行状態、待ち状態、強制待ち状態の5つがある。
2. プロセスには生成時に0~255の優先度が与えられ、この優先度はメインタスクの優先度となる。また、サブタスクにもタスク生成時に優先度が与えられる。
3. 各タスクは優先度順にスケジューリングされ、同一優先度の場合は、ラウンドロビンで平等にスケジューリングされる。
4. プロセスIDは正の整数値であり、プロセスの優先度に基づいて割りつけられる。

問 1 8

T-Kernel (μ ITRON)のディスパッチ禁止／解除に関する以下の説明のうち、正しいものをすべて選べ。

※ μ ITRONの場合は () 内のサービスコールを呼び出すものとして解答せよ。

選択肢

1. tk_dis_dsp (dis_dsp) から tk_ena_dsp (ena_dsp) が実行されるまでの間は、タスクのディスパッチは禁止される。
2. ディスパッチ禁止状態の間は、割込みハンドラからの復帰時のディスパッチも禁止される。
3. ディスパッチ禁止状態で自タスクを待ち状態に遷移する可能性のあるシステムコールを発行した場合でも、実際に待ち状態に遷移しなければエラーにはならない。
4. ディスパッチ禁止／許可のシステムコールは、それぞれ別のタスクから呼び出す必要がある。

問 1 9

μ ITRONのデータキューに関する以下の説明のうち、誤っているものを一つ選べ。

選択肢

1. データキューは、1 ワードのメッセージを受け渡しすることで、同期と通信を行うオブジェクトである。データキューでは、データの送信を待つタスクの待ち行列とデータの受信を待つ待ち行列の二つの待ち行列を持つ。
2. データキューでは、送信されたデータを格納するためのデータキュー領域を持つ。データを送信するタスクは、送信したいデータをデータキューに入れる。データキュー領域に空きがない場合は、エラーが返される。
3. データキューでは、データキュー領域に格納できるデータの数を 0 に設定すると、同期メッセージ機能を実現することができる。
4. データキューで送受信されるデータは、整数値であっても、送信側と受信側で共有しているメモリ上に置かれたメッセージの先頭番地であってもよい。

問 2 0

T-Kernel Standard Extension のメールボックス機能に関する以下の説明のうち、正しいものを一つ選べ。

選択肢

1. tkse_cre_mbx で作成したメールボックス ID を用いて、tk_rcv_mbx などの T-Kernel ネイティブのシステムコールを呼び出すことができる。
2. メールボックスの機能を用いてプロセス間の通信を行うことができる。
3. メールボックスを用いたデータ通信では、メッセージの内容が全てコピーされるため、データの送受信に比較的時間がかかる。
4. メールボックスを生成したプロセスに所属しないタスクは、メールボックスからの受信を行うことはできない。

出題区分2 (5問 各8点)

問21

μ ITRON を利用して書かれた次ページ以降のプログラムを動作させるものとする。

このとき、下記の設問1、設問2に解答せよ。

ただし、プログラムは以下の条件で動作するものとする。

- ・最初に、初期設定をするタスク `create_object` が優先度1で実行される。
- ・ `create_object` 中のすべてのシステムコールは正常に終了するものとする。
- ・ `printf` 関数の処理にかかる時間は無視できる程短いものとする。
- ・ プログラムにないオブジェクトやハンドラは生成されていない。

※ T-Kernel の場合は、以下を前提として解答せよ。

- ・ μ ITRON の `slp_tsk()` は T-Kernel の `tk_slp_tsk(TMO_FEVR)` に相当する。
- ・ μ ITRON の `tslp_tsk(msec)` は T-Kernel の `tk_slp_tsk(msec)` に相当する。
- ・ μ ITRON の `cre_sem`、`wai_sem`、`sig_sem` の第1引数にはセマフォ ID を指定する。
- ・ μ ITRON の `cre_tsk` の第1引数にはタスク ID を指定する。
- ・ μ ITRON の `wai_sem(id)` は、T-Kernel の `tk_wai_sem(id, 1, TMO_FEVR)` に相当する。
- ・ タスク属性に `TA_ACT` が含まれるとタスクは起動された状態で生成される。
- ・ システムコールの名称は `cre_tsk` を `tk_cre_tsk` のように読み替える。

【設問1】

プログラムを実行すると、プログラム中の `printf` によるメッセージが表示される。

最初に表示されるメッセージが `task1: Get semaphore` であるとして、4番目に表示されるメッセージとして正しいものを一つ選べ。

選択肢

1. `task1: Release semaphore`
2. `task2: Get semaphore`
3. `task3: Get semaphore`
4. `task4: Get semaphore`

【設問2】

`printf` では `Get semaphore` として「セマフォ資源を獲得した」と表示しているが、実際にはセマフォ資源を獲得できていないタスクが存在する。

セマフォ資源を獲得できていないタスクは以下のいずれか。正しいものを選択肢の中から一つ選べ。

選択肢

1. `task1`
2. `task2`
3. `task3`
4. `task4`

```

void    create_object(VP_INT exinf)
{
    ER    ercd;
    T_CSEM    csem;
    T_CTSK    ctsk;

    csem.sematr    = TA_TFIFO;           /* Create semaphore */
    csem.isemcnt    = 2;
    csem.maxsem    = 2;
    ercd = cre_sem(1, &csem);

    ctsk.tskatr    = TA_HLNG | TA_ACT;   /* Create tasks */
    ctsk.exinf    = 0;
    ctsk.task    = task1;
    ctsk.itskpri    = 5;
    ctsk.stksz    = STACKSZ;
    ctsk.stk    = stack1;
    ercd = cre_tsk(1, &ctsk);

    ctsk.tskatr    = TA_HLNG | TA_ACT;
    ctsk.exinf    = 0;
    ctsk.task    = task2;
    ctsk.itskpri    = 6;
    ctsk.stksz    = STACKSZ;
    ctsk.stk    = stack2;
    ercd = cre_tsk(2, &ctsk);

    ctsk.tskatr    = TA_HLNG | TA_ACT;
    ctsk.exinf    = 0;
    ctsk.task    = task3;
    ctsk.itskpri    = 7;
    ctsk.stksz    = STACKSZ;
    ctsk.stk    = stack3;
    ercd = cre_tsk(3, &ctsk);

    ctsk.tskatr    = TA_HLNG | TA_ACT;
    ctsk.exinf    = 0;
    ctsk.task    = task4;
    ctsk.itskpri    = 8;
    ctsk.stksz    = STACKSZ;
    ctsk.stk    = stack4;
    ercd = cre_tsk(4, &ctsk);

    ext_tsk();
}

```

```

void    task1(VP_INT exinf)
{
    ER        ercd;

    ercd = wai_sem(1);
    printf("task1: Get semaphore\n");
    ercd = tslp_tsk(1000);
    ercd = sig_sem(1);
    printf("task1: Release semaphore\n");
    ercd = slp_tsk();
    /* 略 */
}

void    task2(VP_INT exinf)
{
    ER        ercd;

    ercd = wai_sem(3);
    printf("task2: Get semaphore\n");
    ercd = slp_tsk();
    /* 略 */
}

void    task3(VP_INT exinf)
{
    ER        ercd;

    ercd = wai_sem(1);
    printf("task3: Get semaphore\n");
    ercd = slp_tsk();
    /* 略 */
}

void    task4(VP_INT exinf)
{
    ER        ercd;

    ercd = wai_sem(1);
    printf("task4: Get semaphore\n");
    ercd = slp_tsk();
    /* 略 */
}

```

問 2 2

T-Kernel を利用して、不定期のタイミングで動作する割り込みハンドラ inthdr の累計実行回数を、約 1 分毎に表示するプログラムを作成するものとする。

累計実行回数の表示は tskid_A のタスク ID を持つ task_A で行うものとし、task_A では、約 1 分毎に動作する関数 count を使って割り込みハンドラの累計実行回数を取得する。

このような機能を実現する関数 count のプログラムとして正しいものを、選択肢の中からすべて選べ。

ただし、プログラムは以下の条件で動作するものとする。

- ・ 起床要求キューイング数の最大数は 65535 とする。
- ・ 割り込みの発生回数は、毎秒 10~50 回、毎分 1000~2000 回とする。
- ・ W 型のオーバーフローについては考慮しなくてよい。
- ・ printf の %d では W 型を正常に表示することができる。

※ μ ITRON の場合は、以下を前提として解答せよ。

- ・ T-Kernel の tk_wup_tsk は μ ITRON の wup_tsk、iwup_tsk に相当する。
- ・ T-Kernel の tk_slp_tsk(TMO_FEVR) は μ ITRON の slp_tsk() に相当する。
- ・ サービスコールの名称は tk_dly_tsk を dly_tsk のように読み替える。

```
void    inthdr(UINT dintno)
{
    tk_wup_tsk(tskid_A);
    /* 略：この割り込みに対する割り込み要求をクリアする処理 */
}

W       intr_cnt;

void    task_A(INT stacd, VP exinf)
{
    intr_cnt = 0;
    for(;;) {
        tk_dly_tsk(60000);      /* 自タスクを 1 分間遅延 */
        count();
        printf("割り込み発生回数の合計 = %d\n", intr_cnt);
    }
}
```

選択肢

1.

```
void    count(void)
{
    intr_cnt += tk_can_wup(TSK_SELF);
}
```

2.

```
void    count(void)
{
    T_RTSK  rtsk;

    tk_ref_tsk(TSK_SELF, &rtsk);
    intr_cnt = rtsk.wupcnt;
}
```

3.

```
void    count(void)
{
    T_RTSK  rtsk;
    INT     i;

    tk_ref_tsk(TSK_SELF, &rtsk);
    i = intr_cnt = rtsk.wupcnt;
    while(i-- > 0) {
        tk_slp_tsk(TMO_FEVR);
    }
}
```

4.

```
void    count(void)
{
    T_RTSK  rtsk;

    tk_ref_tsk(TSK_SELF, &rtsk);
    intr_cnt += rtsk.wupcnt;
    while(rtsk.wupcnt-- > 0) {
        tk_slp_tsk(TMO_FEVR);
    }
}
```

問 2 3

μITRON を利用して書かれた次ページのプログラムを動作させるものとする。

このプログラムの割り込み処理は全体として最大何バイトのスタックを消費するか。消費されるスタックの最大サイズを選択肢の中から一つ選べ。

ただし、プログラムは以下の条件で動作するものとする。

- ・ 各サービスコール、関数のスタックサイズの使用量は以下のとおりとする。

関数名	スタック消費量
isig_sem	30 バイト
iact_tsk	20 バイト
func1	40 バイト
func2	20 バイト

- ・ 割り込みハンドラ内での多重割り込みは許可されている。
- ・ すべての割り込みは非同期に発生し、割り込みは 7 段階の割り込み優先レベルによって制御される。
- ・ 割り込みのレベルは、値の小さいほうが優先度が高く、同一のレベルの場合は割り込みの受付は遅延される。
- ・ 割り込みハンドラと割り込み優先レベルの対応は以下のとおりとする。

割り込みハンドラ	割り込み優先レベル
inh1	2
inh2	3
inh3	3

- ・ サービスコール処理内ではすべての割り込みを禁止している。
- ・ 割り込みハンドラで使用するスタック、サービスコールで使用するスタックは共に同一の連続したスタック領域を使用するものとする。
- ・ 省略した処理ではスタックの消費はなく、関数、サービスコール呼び出しの部分以外についてもスタック消費はないものとする。

※ T-Kernel の場合は、以下を前提として解答せよ。

- ・ μITRON の isig_sem は T-Kernel の tk_sig_sem に相当する。
- ・ μITRON の iact_tsk は、対象タスクを休止状態から実行可能状態に移行させるサービスコールである。

選択肢

1. 90 バイト
2. 40 バイト
3. 60 バイト
4. 50 バイト

```
void    inh1(void)
{
    /* 略 */
    iact_tsk(ID_task1);
    /* 略 */
}

void    inh2(void)
{
    /* 略 */
    func2();
    /* 略 */
    isig_sem(ID_SEM1);
    /* 略 */
}

void    inh3(void)
{
    /* 略 */
    func1();
    /* 略 */
}
```

問 2 4

T-Kernel を利用して書かれた次ページのプログラムを動作させるものとする。

このとき、周期ハンドラからの復帰直後に実行されるタスクは以下のいずれか。選択肢の中から一つ選べ。

ただし、プログラムは以下の条件で動作するものとする。

- ・ プログラムで使用している変数などは別途宣言されている。
- ・ 各タスクのタスク ID、初期状態、優先度は以下のとおりとする。

タスク	タスク ID	初期状態	優先度
タスク 1	tsk1	実行可能状態	1
タスク 2	tsk2	実行可能状態	1
タスク 3	tsk3	実行可能状態	2
タスク 4	tsk4	実行可能状態	2

- ・ 周期ハンドラ `cychdr` に関するハンドラ生成情報などは以下のとおりとする。

生成情報	設定値
周期ハンドラ ID	<code>cyc1</code>
周期起動時間間隔	100ms
周期ハンドラの起動	生成時に起動する
周期起動位相	保存しない

- ・ イベントフラグに関するイベントフラグ生成情報などは以下のとおりとする。

生成情報	設定値
イベントフラグ ID	<code>flg1</code>
ビットパターン初期値	0
イベントフラグ属性	<code>TA_TFIFO TA_WMUL</code>

※ μ ITRON の場合は、以下を前提として解答せよ。

- ・ T-Kernel の `tk_wai_flg` は μ ITRON の `twai_flg` に相当する。
- ・ システムコールの名称は `tk_set_flg` を `set_flg` のように読み替える。

選択肢

1. タスク 1
2. タスク 2
3. タスク 3
4. タスク 4

```

void task1(INT stacd, VP exinf)
{
    tk_wai_flg(flg1, 0x0008, TWF_ANDW, &flgptn, TMO_FEVR);
    /* 略：タスク 1 用の処理 */
    tk_ext_tsk();
}

void task2(INT stacd, VP exinf)
{
    tk_wai_flg(flg1, 0x0007, TWF_ANDW, &flgptn, TMO_FEVR);
    /* 略：タスク 2 用の処理 */
    tk_ext_tsk();
}

void task3(INT stacd, VP exinf)
{
    tk_dis_dsp();
    /* ここで周期ハンドラが起動する */
    /* 略：タスク 3 用の処理 */
    tk_ena_dsp();
    tk_ext_tsk();
}

void task4(INT stacd, VP exinf)
{
    tk_wai_flg(flg1, 0x0007, TWF_ORW | TWF_CLR, &flgptn, TMO_FEVR);
    /* 略：タスク 4 用の処理 */
    tk_ext_tsk();
}

void cychdr(VP exinf)          /* 周期ハンドラの処理 */
{
    tk_set_flg(flg1, 0x0008);
    tk_set_flg(flg1, 0x0007);
    return;
}

```

問 2 5

T-Kernel を利用して書かれた次ページのプログラムを動作させるものとする。

このとき、下記の設問 1、設問 2 に解答せよ。

ただし、プログラムは以下の条件で動作するものとする。

- ・ 起動時には `task_init()` が優先度 5 のタスクとして実行される。
- ・ `task_init()` 中のすべてのシステムコールは正常に終了する。
- ・ T_CTSK 構造体のメンバは次の通りとし、これ以外のメンバの記載は省略している。

```
typedef struct t_ctsk {
    VP    exinf;        /* 拡張情報 */
    ATR   tskatr;      /* タスク属性 */
    FP    task;        /* タスク起動アドレス */
    PRI   itskpri;     /* タスク起動時優先度 */
    /* これ以下のメンバの記載は省略 */
} T_CTSK;
```

- ・ T_CPOR 構造体のメンバは次の通りとし、これ以外のメンバの記載は省略している。

```
typedef struct t_cpor {
    VP    exinf;        /* 拡張情報 */
    ATR   poratr;      /* ランデブポート属性 */
    INT   maxcmsz;     /* 呼出メッセージの最大長(バイト) */
    INT   maxrmsz;     /* 返答メッセージの最大長(バイト) */
    /* これ以下のメンバの記載は省略 */
} T_CPOR;
```

- ・ `tk_dly_tsk` による遅延時間は十分に正確なものとし、遅延時間の不正確さにより以下の問いに対する解答に影響を与えることはないものとする。
- ・ `tk_dly_tsk` 以外のシステムコールの実行時間やその他のプログラムの実行時間は十分に短いものとし、これらの時間が以下の問いに対する解答に影響を与えることはないものとする。

※ μ ITRON の場合は、以下を前提として解答せよ。

- ・ T-Kernel の `tk_cre_por` は μ ITRON の `acre_por` に相当する。
- ・ T-Kernel の `tk_cre_tsk` は μ ITRON の `acre_tsk` に相当する。
- ・ T-Kernel の `tk_acp_por` は μ ITRON の `tacp_por` に相当する。
- ・ T-Kernel の `tk_cal_por` は μ ITRON の `tcal_por` に相当する。
- ・ T-Kernel では `tk_dly_tsk` の引数をミリ秒単位で指定する。
- ・ サービスコールの名称は `tk_ext_tsk` を `ext_tsk` のように読み替える。

【設問1】

次ページのプログラムを実行してから 10 秒後の `result` の数値を解答欄にマークしなさい。

【設問2】

次ページのプログラムを実行してから 30 秒後の `result` の数値を解答欄にマークしなさい。

```

#define MAX_CAL      8
#define MAX_ACP      3
ID      tskid_cal[MAX_CAL + 1];
ID      tskid_acp[MAX_ACP];
ID      port_id;
INT     result;

void     task_init(INT stacd, VP exinf)    /* 初期化タスク(優先度 5) */
{
    T_CTSK  ctsk_acp = {NULL, TA_HLNG, task_acp, 81, /* 他のメンバは省略 */ };
    T_CTSK  ctsk_cal = {NULL, TA_HLNG, task_cal, 82, /* 他のメンバは省略 */ };
    T_CPOR  cpor = {NULL, TA_TFIFO, 16, 16};
    INT     i;

    result = 9;
    port_id = tk_cre_por(&cpor);

    for(i = 0; i < MAX_ACP; i++){
        tskid_acp[i] = tk_cre_tsk(&ctsk_acp);
        tk_sta_tsk(tskid_acp[i], 0);
    }
    for(i = 1; i <= MAX_CAL; i++){
        tskid_cal[i] = tk_cre_tsk(&ctsk_cal);
        tk_sta_tsk(tskid_cal[i], i);
    }
    tk_exd_tsk();
}

void     task_acp(INT stacd, VP exinf)
{
    B       msg[4];
    RNO     rdvno;

    for(;;){
        tk_acp_por(port_id, 0x01, &rdvno, msg, TMO_FEVR);
        tk_dly_tsk(msg[0] * 1000);
        tk_rpl_rdv(rdvno, msg, 1);
        tk_dly_tsk(1000);
    }
}

void     task_cal(INT stacd, VP exinf)
{
    B       msg[4];

    tk_dly_tsk(500);
    for(;;){
        msg[0] = stacd;
        tk_cal_por(port_id, 0x01, msg, 1, TMO_FEVR);
        result = stacd;
        tk_dly_tsk(1000);
    }
}

```

(計算用紙)

C. 解答上の注意

1. 氏名欄

氏名欄には、氏名、フリガナを記入しなさい。

2. 試験場欄

試験場欄は、受験している試験場所に○が付けられていることを確認しなさい。

(記入例) 東京で受験する場合

東京・大阪・他

3. 受験日欄

受験日欄には、今日の年月日が記入されていることを確認しなさい。

(記入例) 2010年6月5日の場合

20	10	年	06	月	05	日
----	----	---	----	---	----	---

4. 受験番号欄

解答用紙の受験番号欄に受験番号が正しく記入され、さらにその下のマーク欄に正しくマークされていることを確認しなさい。

正しくマークされていない場合は、採点できないことがあります。

5. B または HB の黒鉛筆または黒色シャープペンシルを使用して正しく記入、マークしなさい。訂正の場合は、あとが残らないように消しゴムできれいに消し、消しくずを残さないようにしなさい。

6. 解答用紙は光学式読取装置で自動処理しますので、解答用紙のマーク例にしたがって正しくマークしなさい。

7. 問題文において「選択肢から一つ選べ」もしくは「一つずつ選べ」と指示されている場合は、解答欄に1つだけマークしなさい。

(記入例) 3を選択する場合

①	②	●	④
---	---	---	---

8. 問題文において「選択肢からすべて選べ」と指示されている場合は、解答欄に1つ以上マークしなさい。

(記入例) 2と4を選択する場合

①	●	③	●
---	---	---	---

9. 問題文において数値を選択することが指示されている場合は、解答欄では1つの桁に1つだけマークしなさい。

(記入例) 6を選択する場合

①	②	③	④	⑤	●	⑦	⑧	⑨	⑩
---	---	---	---	---	---	---	---	---	---

表紙に戻る場合は、この問題冊子を右側から裏返しなさい。

試験開始の合図があるまで、この問題冊子を開いてはいけません。