最終更新日：2009年2月25日

FMPカーネルテストプログラムマニュアル

名古屋大学 大学院情報科学研究科
附属組込み研究センター

テストプログラムの概要

本テストプログラムには，大きく2種類のテストが含まれている．
テスト項目については，今後拡充していく予定である

1コア動作テスト

FMPを1コアで動作させ，カーネルの基本機能をテストする．

- make_runnable()/make_non_runnable()の網羅的なテスト

2コア動作テスト

2コアでFMPを動作させ，FMP独自APIの基本動作を確認する

- mig_tsk テスト
- mact_tsk テスト

テストプログラムのファイル構成

1コア動作テスト

- make_runnable()/make_non_runnable()の網羅的なテスト
  - test_1core.c/h/cfg

2コア動作テスト

- mig_tsk()テスト
  - test_mig_tsk1.c/h/cfg
  - test_mig_tsk2.c/h/cfg
- mact_tsk()テスト
  - test_mact_tsk1.c/h/cfg
  - test_mact_tsk2.c/h/cfg
  - test_mact_tsk3.c/h/cfg

テスト用ライブラリプログラム

- test_lib.c/h

テストプログラムの実行方法

- プロジェクトディレクトリの作成
  - Makefileはsample1付属のものをベースにする
  - コア数はテスト毎に異なるため，個々のテストに従う
- コンパイル対象のファイルのコピー
  - テスト用ライブラリ（test_lib.c/test_lib.h）
  - テスト毎のファイル（h/c/cfg）
- Makeファイルの編集
  - APPNAMEをテスト毎のファイル名とする
  - APPL_COBJS に test_lib.o を追加
- ビルドと実行
  - チェックポイントを全て通過すればテストをパスしたとみなす

test_lib.c/hの拡張内容

- ASP付属のファイルをマルチコア用に拡張
  - チェックポイントをコアごとに管理するように拡張

テストプログラム

```
task(){
    chech_point(3)
}
```

＜正解＞
check_pointをcpu2で3番目に通過する

test_lib.c

コアごとの変数を配列に格納

```
static ID    check_count[TNUM_PRC] = {0u};

void
check_point(uint_t count)
{
```

コアごとに予定された順に，check_pointを通過したかを判定

get_pid()により自CPU№.を取得し，check_count[]に格納された自コアの現在のチェックポイント番号と，受け取ったcountを比較して，正しくcheck_pointを通過したか判定する

```
}
```

チェックポイントは，コアごとに管理し，正しいコアで，正しい順に通過することを確認

1コア動作テスト

- 内容
  - 1コアで動作させ，make_runnable() /make_non_runnable()の網羅的なテストを行う
- ファイル
  - test_1core.cfg / test_1core.h /test_1core.cfg
- テストパス条件
  - チェックポイントを全て（40個）通過すればテストパス

```
Processor 1 start.
System logging task is started on port 1.
Check point 1 passed.
Check point 2 passed.
Check point 3 passed.
Check point 4 passed.
Check point 5 passed.
Check point 6 passed.
Check point 7 passed.
Check point 8 passed.
Check point 9 passed.
・・・・
```

2コア動作テスト : mig_tskテスト(1)

- 内容
  - 2コアで動作させ，mig_tsk のパターン1と2をテストする
- ファイル
  - test_mig_tsk1.cfg / test_mig_tsk1.h / test_mig_tsk1.c
- テストパス条件
  - 2コア共にチェックポイントを通過すればOK
    - コア1 : 10個
    - コア2 : 11個

2コア動作テスト : mig_tskテスト(2)

- 内容
  - 2コアで動作させ，mig_tsk のパターン3をテストする
- ファイル
  - test_mig_tsk2.cfg / test_mig_tsk2.h / test_mig_tsk2.c
- テストパス条件
  - 2コア共にチェックポイントを通過すればOK
    - コア1 :　9個
    - コア2 : 12個

2コア動作テスト : mact_tskテスト(1)

- 内容
  - 2コアで動作させ，mact_tsk のパターン1のaをテストする
- ファイル
  - test_mact_tsk1.cfg / test_mact_tsk1.h / test_mact_tsk1.c
- テストパス条件
  - 2コア共にチェックポイントを通過すればOK
    - コア1 : 5個
    - コア2 : 8個

2コア動作テスト : mact_tskテスト(2)

- 内容
  - 2コアで動作させ，mact_tsk のパターン1のbをテストする
- ファイル
  - test_mact_tsk2.cfg / test_mact_tsk2.h / test_mact_tsk2.c
- テストパス条件
  - 2コア共にチェックポイントを通過すればOK
    - コア1 : 8個
    - コア2 : なし

2コア動作テスト : mact_tskテスト(3)

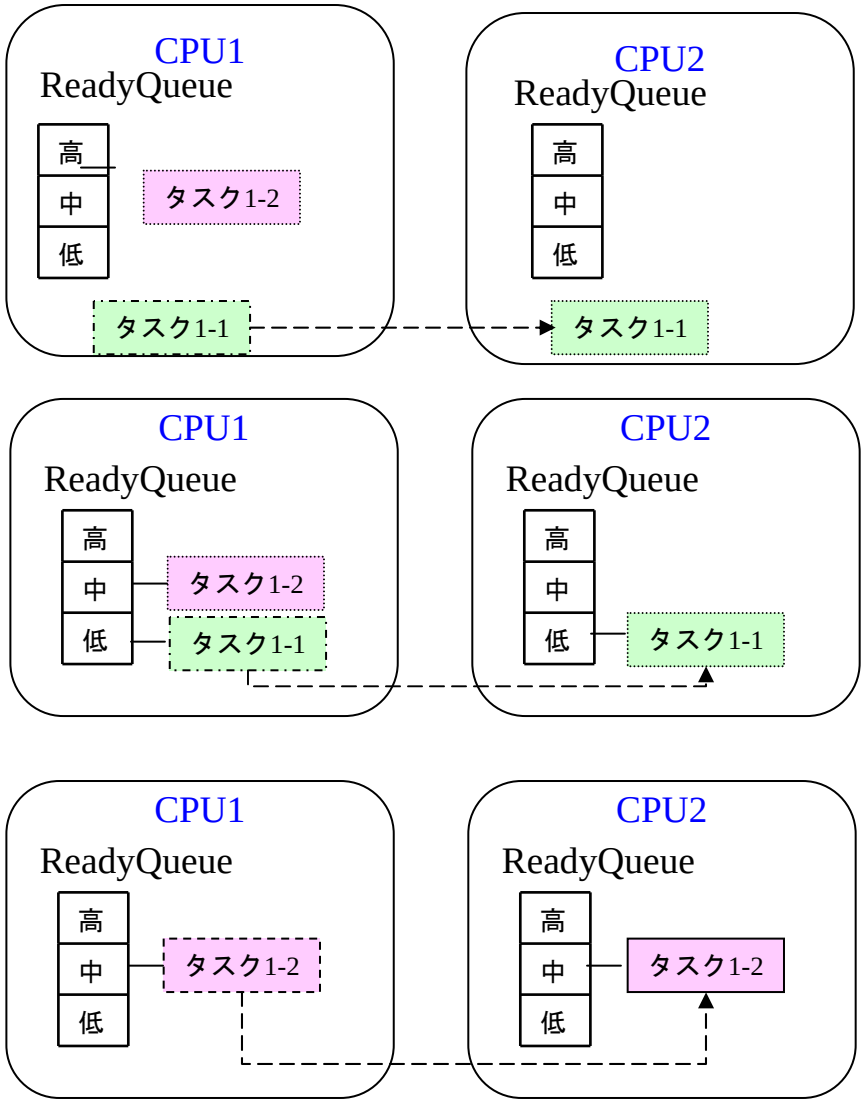
- 内容
  - 2コアで動作させ，mact_tsk のパターン1のbをテストする
- ファイル
  - test_mact_tsk3.cfg / test_mact_tsk3.h / test_mact_tsk3.c
- テストパス条件
  - 2コア共にチェックポイントを通過すればOK
    - コア1 : 13個
    - コア2 :　5個

テスト内容詳細

mig_tskテスト(1)(2) ：テスト概要

マイグレートする対象タスクの状態により，パターン1,2,3に分類

- パターン1
  - 対象タスクが休止状態
    - ReadyQueueにつながっていない
- パターン2
  - 対象タスクが実行可能状態
    - ReadyQueueにつながっている他をタスクを移動
- パターン3
  - 対象タスクが実行状態
    - 自分自身が移動する



mig_tskテスト(1)(2) ：テストパターン

- パターン2，パターン3に関しては，移動後のReadyQueueの状態で，さらに場合分けし，以下のテストパターンを抽出

対象タスクの状態 ＼ 起動の結果		最高優先度になる	実行状態のタスクと同優先度になる	実行状態のタスクより低優先度になる	レディキューが空の所へ移動する
パターン1	休止状態（レディキューにつながっていない）	1-A			
パターン2	ready状態（レディキューにつながっている）	1-B	1-C	1-D	1-E
パターン3	自分自身（running状態）	1-F	1-G	1-H	1-I

mig_taskテスト(1) : 詳細シーケンス

- パターン1と2をテスト

	コア１						コア2					
	コア1　レディキュー	RUNタスク	check_point	テストパターン	サービスコール		コア2レディキュー	RUNタスク	check_point	サービスコール	
1-5	1-1 1-3 / 1-4 1-6 / 1-7 1-8 1-9										
		1-1	①	1-A	mig_tsk(1-5,2)		1-3				
			②	1-E	mig_tsk(1-3,2)	1-5					
					フラグ8待ち			1-3	①	フラグ8セット	
											同期1　フラグ8
	1-1 / 1-6 / 1-8 1-9	1-1	③	1-D	mig_tsk(1-4,2)		1-3 / 1-4 / 1-7				
			④	1-D	mig_tsk(1-7,2)						
			⑤		フラグ1セット			1-3		フラグ1待ち	
											同期2　フラグ1
		1-1			フラグ2待ち		1-3 / 1-4 1-5 / 1-7	1-3	②	act_tsk(1-5)	
									③	slp_tsk	
								1-4	④	slp_tsk	
								1-5	⑤	slp_tsk	
								1-7	⑥	フラグ2セット	
							1-7				
											同期3　フラグ2
	1-1 / 1-8 1-9							1-7		フラグ4待ち	
		1-1	⑥	1-B	mig_tsk(1-6,2)						
							1-6 / 1-7 1-9	1-6	⑦	フラグ4セット	
									⑧	slp_tsk	
			⑦		slp_tsk						
		1-8	⑧		フラグ10待ち			1-7	⑨	フラグ10セット	
			⑨	1-C	mig_tsk(1-9,2)				⑩	slp_tsk	
			⑩		テスト終了			1-9	⑪	テスト終了	

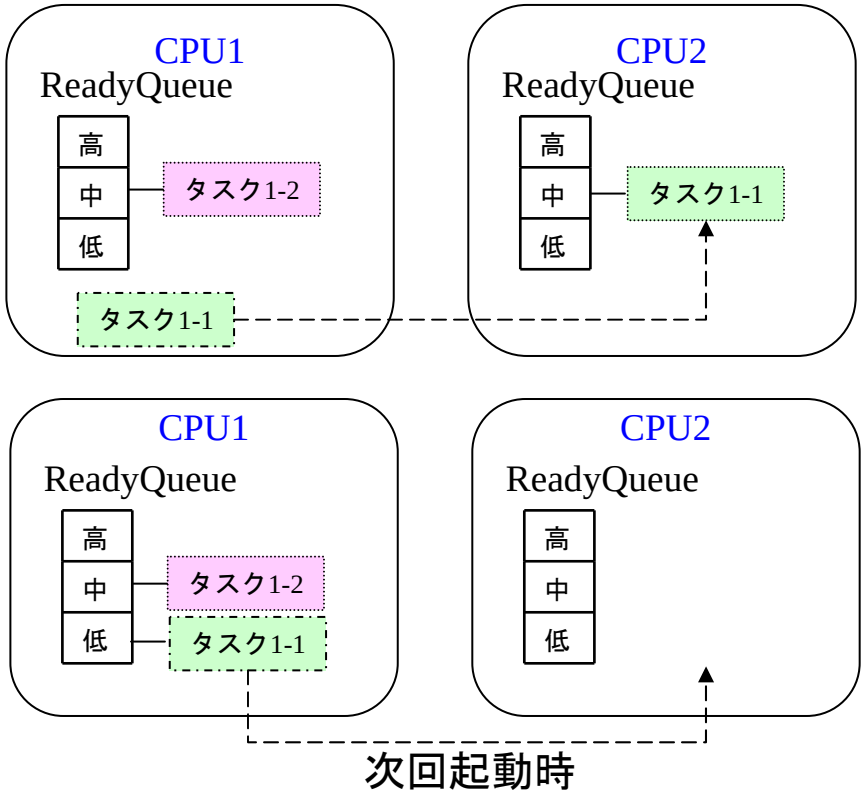
mig_taskテスト(2) : 詳細シーケンス

- パターン3をテスト

コア1 レディキュー	RUNタスク	check_point	テストパターン	サービスコール	コア2レディキュー	RUNタスク	check_point	サービスコール	
	コア 1					コア2			
1-1 / 1-4 / 1-7 1-9									
1-1 / 1-4 / 1-7 1-9	1-1	①	1-I	mig_tsk(1-1,2)	1-1	1-1	①	act_tsk(2-2)	
							②	act_tsk(2-3)	
							③	slp_tsk	
1-4 / 1-7 1-9	1-4	②		フラグ2セット	2-2 2-3	2-2	④	フラグ2待ち	
									同期1 フラグ2
	1-4			フラグ10待ち	2-3	2-3	⑤	slp_tsk	
							⑥	フラグ10セット	
									同期2 フラグ10
	1-4	③	1-G	mig_tsk(1-4,2)	2-3 1-4	2-3		フラグ4待ち	
1-7 1-9	1-7	④		フラグ4セット					
									同期3 フラグ4
1-6 / 1-9	1-7	⑤	1-H	mig_tsk(1-7,2)		2-3	⑦	slp_tsk	
	1-9	⑥		act_tsk(1-6)	1-7	1-4	⑧	slp_tsk()	
	1-6	⑦		フラグ1待ち		1-7	⑨	フラグ1セット	
									同期4 フラグ1
	1-6	⑧	1-F	mig_tsk(1-6,2)		1-6	⑩	フラグ8セット	
				フラグ8セット	1-6		⑪	slp_tsk	
	1-9	⑨		テスト終了	1-7	1-7		フラグ8待ち	
							⑫	テスト終了	

mact_tskテスト(1)(2)(3) : テスト概要

- 対象タスクの状態により分類
  - パターン1
    - 対象タスクが休止状態
      - ReadyQueueにつなぐ
  - パターン2
    - 対象タスクが休止状態以外
      - キューイング数の操作のみ



- 対象タスクの所属により分類
  - パターンa
    - 自プロセッサ所属のタスクを他プロセッサへ
  - パターンb
    - 他プロセッサ所属のタスクを自プロセッサへ
  - パターンc
    - 他プロセッサ所属のタスクを他プロセッサへ

mact_tskテスト(1)(2)(3) : テストパターン

- パターン1
  - 対象タスクが起動するReadyQueueの状態で，さらに場合分け
- パターン1，2とパターンa,b,cを組み合わせ，テストパターンを抽出

対象タスクの状態	起動の結果 / 対象タスクの所属	最高優先度	実行状態のタスクと同優先度	実行状態のタスクより低い優先度	レディキューが空
パターン1 休止状態 （レディキューへつなぐ作業発生）	パターンa 自→他	2-A	2-B	2-C	2-D
	パターンb 他→自	2-E	2-F	2-G	
	パターンc 他→他	実施せず	実施せず	実施せず	実施せず
パターン2 休止状態以外 （レディキューへつなぐ作業なし）	パターンa 自→他	2-H			
	パターンb 他→自	2-I			
	パターンc 他→他	実施せず	実施せず	実施せず	実施せず

mact_tskテスト(1)：詳細シーケンス

- 対象タスクが，休止状態，自プロセッサのタスクを他プロセッサへ

コア１						コア2				
コア1　レディキュー	RUNタスク	check_point	テストパターン	サービスコール	コア2レディキュー	RUNタスク	check_point	サービスコール	
1−4									
	1−4	①	2−D	mact_tsk(1−1,2) フラグ１待ち	1−1	1−1	①		
					2−2		②	act_tsk(2−2)	
							③	slp_tsk	
						2−2	④	フラグ１セット	
									同期１　フラグ１
		②	2−A	mact_tsk(1−2,2)	1−2	2−2		フラグ２待ち	
		③	2−B	mact_tsk(1−5,2)	2−2　1−5	1−2	⑤	フラグ２セット	
		④	2−C	mact_tsk(1−7,2)	1−7			slp_tsk	
		⑤		テスト終了		2−2	⑥	slp_tsk	
						1−5	⑦	slp_tsk	
						1−7	⑧	テスト終了	

mact_tskテスト(2)：詳細シーケンス

- 対象タスクが，休止状態，他プロセッサのタスクを自プロセッサへ

コア１						コア2				
コア1　レディキュー		RUNタスク	check_point	テストパターン	サービスコール	コア2レディキュー	RUNタスク	check_point	サービスコール	
	1-4									
	2-1	1-4	①	2-E	mact_tsk(2-1,1)					
	1-4	2-1	②		フラグ1セット					
			③		slp_tsk					
		1-4			フラグ１待ち					
			④	2-F	mact_tsk(2-4)					
	1-4 2-4		⑤	2-G	mact_tsk(2-7)					
	2-7		⑥		slp_tsk					
		2-4	⑦		slp_tsk					
		2-7	⑧		テスト終了					

mact_tskテスト(3)：詳細シーケンス

- 対象タスクが，起床待ち

	コア１				コア2				
コア1　レディキュー	RUNタスク	check_point	テストパターン	サービスコール	コア2レディキュー	RUNタスク	check_point	サービスコール	
1-4									
	1-4	①		mact_tsk(1-5,2)	1-5	1-5	①	フラグ１セット slp_tsk	
	1-4	②	2-H	mact_tsk(1-5,2) フラグ１待ち					
									同期1　フラグ１
		③		wup_tsk(1-5) フラグ２待ち			② ③	1回目の実行終了 2回目起動 フラグ２セット slp_tsk	
									同期2　フラグ２
2-1 1-4	1-4 2-1 1-4 2-1 1-4 2-1	④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ ⑫ ⑬	2-I	wup_tsk(1-5) mact_tsk(2-1,1) slp_tsk mact_tsk(2-1,1) wup_tsk(2-1) 1回目終了 2回目起動 slp_tsk wup_tsk(2-1) slp_tsk テスト終了			④ ⑤	 テスト終了	