セマフォ獲得問題

プログラム

|  |
| --- |
| #include<stdio.h>  #include<string.h>  //キューが外れている  #define READY 1  //キューがつながってる  #define RUN 2  //待機  #define WAIT 3  //プロトタイプ宣言  int funcA(void);  int funcB(void);  int funcC(void);  //初期化関数  void set(void);  int askA(void);  //セマフォ管理インクリメント  int get\_sem(int sem);  //セマフォ管理インクリメント  int rel\_sem(int sem);  //命令カウンター  int cotl(void);  //タスクコントローラー  struct TCB{      //id      int id;      //状態      int state;      //次へのポインター      struct TCB \*next;      //関数ポインタ      int (\*task)();  }tcb[4];  //デキューイング  void put(struct TCB \*puttcb);  //エンキューイング  void get(struct TCB \*gettcb,struct TCB \*getqtcb);  //分岐ポインタ  struct TCB \*tcb\_branch[3];  //  struct TCB \*btcb=&tcb[0];  int main(void){      //実行ポインタ      struct TCB \*nowtcb;      //実行ポインタと最後のポインタの共用      struct TCB \*qtcb;      //実行ポインタ記憶用      struct TCB \*dqtcb;      //実行カウンタ      int count;      //リスト番号      int counter;      //セマフォ（分岐判断)      int semapho;      set();//初期化      nowtcb=&tcb[0];//開始ポインタ      tcb\_branch[0]=&tcb[0];//メインタスク（TaskAtoTaskC）先頭ポインタ      tcb\_branch[1]=&tcb[3];//割り込みポインタ（askA）      for(count=0;count<15;count++){//実行カウンタ0~14（15回）          tcb\_branch[1]->next=tcb\_branch[0];          semapho=cotl();//命令呼び出し          qtcb=nowtcb;//エンドポインタ更新          dqtcb=nowtcb;//ポインタ記憶          if(semapho==0){              if(tcb\_branch[1]==nowtcb){                  nowtcb=nowtcb->next;//askA飛ばす                  put(qtcb);//askAポインタの連結削る                  btcb->next=nowtcb;//askA移動防止              }          }          nowtcb->task();//タスク実行          if(semapho==0){//分岐なし              counter=0;//リスト番号初期化              printf("semaphone=0\n");//セマフォ確認デバック用              nowtcb=nowtcb->next;//次のポインタ（タスク）へ移動              printf("NULLcheack\n");//タスク割り込み確認防止の実行確認デバック用              while(NULL!=qtcb->next&&dqtcb!=qtcb->next){//エンドポインタ探索                  qtcb=qtcb->next;//次のポインタ移動                  counter++;//リスト数確認              }              printf("NULL clear\n");              if(counter>1){//リストの時のキューイング等                  put(dqtcb);//デキューイング                  get(dqtcb,qtcb);//エンキューイング                  btcb=dqtcb;//              }          }else if(semapho>0){//タスク切り替え              counter=0;              qtcb=nowtcb;              printf("semaphone=1\n");              while(NULL!=qtcb->next&&dqtcb!=qtcb->next){                  printf("cheack\n");                  qtcb=qtcb->next;                  counter++;                  printf("counter=%d\n",counter);              }              printf("chack end\ncounter=%d\n",counter);              if(counter>=1){//リストと判断したら                  tcb\_branch[0]=dqtcb->next;//メインタスクのアドレス                  tcb\_branch[2]=tcb\_branch[1];//                  dqtcb->state=WAIT;                  put(dqtcb);                  get(dqtcb,qtcb);              }else{                  tcb\_branch[2]=tcb\_branch[0];                  dqtcb->state=WAIT;                  put(dqtcb);              }              nowtcb->next=tcb\_branch[2];              nowtcb=nowtcb->next;          }      }  }  void put(struct TCB \*puttcb){      printf("%d\n",puttcb->state);      puttcb->next=NULL;//nextpointer is NULL      puttcb->state=READY;      printf("%d\r\n",tcb->state);  }  void get(struct TCB \*gettcb,struct TCB \*getqtcb){      getqtcb->next=gettcb;//get pointer      gettcb->state=RUN;      printf("%d\r\n",gettcb->state);  }  void set(void){      tcb[0].id=0;      tcb[0].state=RUN;      tcb[0].task=funcA;      tcb[0].next=&tcb[1];      tcb[1].id=1;      tcb[1].state=RUN;      tcb[1].task=funcB;      tcb[1].next=&tcb[2];      tcb[2].id=2;      tcb[2].state=RUN;      tcb[2].task=funcC;      tcb[2].next=NULL;      tcb[3].id=2;      tcb[3].state=WAIT;      tcb[3].task=askA;      tcb[3].next=NULL;  }  int askA(void){      printf("askA\n");      return 0;  }  int cotl(void){      static int prg=0;      static int regsem=2;      static int full=0;      if(prg==0){          printf("get\n");          regsem=rel\_sem(regsem);          prg++;          return 0;      }else if(prg==1){          printf("get\n");          full=get\_sem(full);          prg++;          return full;      }else if(prg==2){            prg++;          return 1;      }else if(prg==3){          prg++;          return 0;      }else if(prg==4){          prg++;          return 0;      }else if(prg==5){          prg++;          return 1;      }else if(prg==6){          prg++;          return 1;      }else if(prg==7){          return 1;      }else if(prg==8){          return 0;      }else if(prg==9){          return 0;      }else{          return 0;      }  }  int funcA(void){      printf("funcA()\r\n");      printf("7bjk1101\r\n");      return 0;  }  int funcB(void){      printf("funcB()\r\n");      printf("inoue shunnsuke\r\n");      return 0;  }  int funcC(void){      printf("funcC()\r\n");      printf("2019/10/17\r\n");      return 0;  }  int get\_sem(int sem){      sem++;      return sem;  }  int rel\_sem(int sem){      sem--;      return sem;  } |

実行結果

|  |
| --- |
| get  funcA()  7bjk1101  semaphone=0  NULLcheack  NULL clear  2  1  2  get  funcB()  inoue shunnsuke  semaphone=1  cheack  counter=1  cheack  counter=2  chack end  counter=2  3  2  2  askA  semaphone=1  cheack  counter=1  cheack  counter=2  cheack  counter=3  chack end  counter=3  3  2  2  2  2  funcC()  2019/10/17  semaphone=0  NULLcheack  NULL clear  funcA()  7bjk1101  semaphone=0  NULLcheack  NULL clear  funcC()  2019/10/17  semaphone=1  cheack  counter=1  chack end  counter=1  3  2  2  askA  semaphone=1  cheack  counter=1  cheack  counter=2  chack end  counter=2  3  2  2  askA  semaphone=1  cheack  counter=1  cheack  counter=2  chack end  counter=2  3  2  2  askA  semaphone=1  cheack  counter=1  cheack  counter=2  chack end  counter=2  3  2  2  askA  semaphone=1  cheack  counter=1  cheack  counter=2  chack end  counter=2  3  2  2  askA  semaphone=1  cheack  counter=1  cheack  counter=2  chack end  counter=2  3  2  2  askA  semaphone=1  cheack  counter=1  cheack  counter=2  chack end  counter=2  3  2  2  askA  semaphone=1  cheack  counter=1  cheack  counter=2  chack end  counter=2  3  2  2  askA  semaphone=1  cheack  counter=1  cheack  counter=2  chack end  counter=2  3  2  2  askA  semaphone=1  cheack  counter=1  cheack  counter=2  chack end  counter=2  3  2  2 |