〔公　開〕

ＴＲ－Ｃ－００６６

通信ソフトウェアの自動作成実験

増田　茂則

Shigenori Masuda

１９９１．８．１

ＡＴＲ通信システム研究所

通信ソフトウエア自動作成実験

増田茂則

1991年8月1日

もくじ

1　はじめに

本報告は、筆者が平成３年７月８日から８月２日までの学外実習の検討結果をまとめたものである。

ATR 通信システム研究所では、通信サービスの動作を記述する手法として、STR（State Transition Rule）記述を開発し、サービス間の矛盾の検出、標準仕様記述言語 SDL（Functional Specification and Description Language）への変換、詳細な仕様記述開発のための詳細化設計支援手法などの研究を進めている。

本実習では、この研究の過程で生成される SDL 記述を、UNIX 上で疑似的に実行するシステムの構築を目標とした。

利用することのできるツールとしては、UNIX 上で、SDL をＣに変換する KINDRA と呼ばれるシステムが存在する。しかし、UNIX 上で実行可能なＣプログラムを得るためには、疑似実行システムが如何に動作するかを設計し、この目的システムに合わせたＣプログラムを生成しなくてはならない。

以下では、本実習で前提としている電話交換システムのソフトウエア構成と与えられた SDL 記述の概要を述べた後、UNIX 上の疑似実行システム、UNIX 上の疑似実行システムで走行可能なＣプログラムを生成するための条件、今後の課題などについて述べる。

2　電話交換ソフトウエア概要

電話交換システムのソフトウエア構成として、各電話機毎の制御プロセスだけを仮定する。各制御プロセスは、他電話機の制御プロセスとの信号の送受信及び、自分の管理している電話機との信号の送受信を行う。電話交換システム内には、このようなプロセスがいくつでも存在し得る。

例えば、制御プロセスは次のような動作を行う。

動作１）

制御プロセスがアイドル状態（電話機が使用されていない状態に対応する）のとき、電話機から offhook（受話器上げの動作に対応する）信号を受信すると、電話機に対して、発信音を鳴らすように指示を与える。制御プロセスは、電話機の状態に対応し、発信音受信状態に遷移する。

動作２）

制御プロセスが発信音受信状態の時、電話機からダイヤルを受けると、発信音を止めるように電話機に指示を与え、ダイヤルに対応した電話機の制御プロセスに通信信号を送出する。信号受信側では、アイドル状態でこの信号を受信した場合には、受諾を意味する信号を返送し、電話機に呼出音を鳴らす指示を与えた後、呼出音受信状態に遷移する。また、アイドル状態以外でこの信号を受信した場合には、棄却を意味する信号を返送し、状態遷移は行わない。ダイヤルした電話機を制御するプロセスは、相手から受諾を意味する返信を受信した場合には、電話機に呼び返し音を鳴らすように指示を与え、呼び返し音受信状態に遷移する。一方、棄却を意味する返信を受信した場合には、電話機にビジー音を鳴らすように指示を与え、ビジー音受信状態に遷移する。

3 SDL 記述

前節のような制御プロセスの動作を SDL の形式で記述したのが、本実習の出発点として与えられた SDL 図である。SDL 図は、STATE で表される状態、IPT で表される信号受信、OPT で表される信号送信、TEST で表される判断分岐、及び、TASK（タスク）と呼ばれる一般的な処理を記述できる箱で構成され、その箱には mi-mj（i,j は、整数）という番号が付加されていて、mi はその箱の番号、mj は次へ進む箱の番号を示す。STATE では、例えば、mi-mj,mk,ml と番号が付いていたなら、mj,mk,ml は次へ進む箱の番号を示す。また、TEST では、mi-mj,mk と番号が付いていて、mj は yes の箱へ、mk は no の箱へ進むことを示す。電話機や他プロセスからの信号受信は、SDL 上の信号受信に対応する。また、SDL 上の信号送信は、他プロセスへの信号送信のみが対応し、電話機への信号送出は、タスクとして記述されている。

以下では、SDL 記述についてもう少し詳しく説明する。

STATE の状態名には、次のようなものがある。

- idle(A)：電話に何も起こっていない状態。
- dial-tone(A)：ダイヤル投入可能な状態。
- wait：使用者には認識できない内部状態。
- busy-dial(A,B)：話中の相手にダイヤルしたときの状態。
- howler(A)：howler の状態。
- r-path(A,B),ringback(A,B)：相手の電話がなっている状態。
- path(A,B)：A と B がつながった状態。
- busy(A)：ビジー音が聞こえている状態。
- ringing(A,B)：呼出音がなっている状態。

IPT の信号受信には、次のようなものがある。

- offhook(A)：受話器を取ったという信号受信。
- onhook(A)：受話器を置いたという信号受信。
- digit(A,B)：dial をする信号受信。
- timeover(busy-dial(A,B))：busydial の timeover の信号受信。
- timeover(dial-tone(A))：dialtone の timeover の信号受信。
- timeover(busy(A))：busy の timeover の信号受信。
- sig-onhook:B：相手 (B) が受話器を置いたという信号受信。
- s1:B：呼出中の電話が応答（受話器あげ）した時に、相手（X）に送る応答信号の受信。

- s2:B：端末 B への着信要求信号の受信。
- r1-s1:X：端末 X からの応答信号を棄却する信号受信。
- r2-s1:X：端末 X からの応答信号に対する承認信号受信。
- r1-s2:X：端末 X からの着信要求を承認する信号受信。
- r2-s2:X：端末 X からの着信要求を棄却する信号受信。

OPT の信号送信には、次のようなものがある。

- s1:B：IPT で述べたと同じ信号の送信。
- s2:B：IPT で述べたと同じ信号の送信。
- r1-s1:X：IPT で述べたと同じ信号の送信
- r1-s2:B：IPT で述べたと同じ信号の送信。
- r2-s1:B：IPT で述べたと同じ信号の送信。
- r2-s2:B：IPT で述べたと同じ信号の送信。
- r2-s2:C：IPT で述べたと同じ信号の送信。
- sig-onhook:B：相手 (B) が受話器を置いたという信号送信。

TEST の判断分岐には、次のようなものがある。

- X == wrongno：電話番号間違いかどうかの判断分岐。
- X == A：電話番号の判断分岐。
- X == B：電話番号の判断分岐。

(注) X は、一時的に電話番号を入れる文字。
A は、自分の電話番号を入れる文字。
B は、X に入れた電話番号を貯めておく文字。
wrongno は、電話番号間違いを意味する。

TASK（タスク）の処理には、次のようなものがある。

- B:=X：B に X を代入するように信号送出する。
- terminal(start:dialtone)：dialtone を始めるように信号送出する。
- terminal(start:silent)：tone を止めるように信号送出する。
- terminal(start:busytone)：busytone を始めるように信号送出する。
- terminal(start:howler)：howler を始めるように信号送出する。

- terminal(start:ringback) : ringback を始めるように信号送出する。
- terminal(start:ringing) : ringing を始めるように信号送出する。
- terminal(hunt B) : 相手 (B) までの回線を確保する。
- terminal(release B) : 占有していた回線を解放する。
- terminal(connect B) : 相手 (B) につながったという信号を送出する。
- time-set(dial-tone(A):20) : dialtone を 20 秒間 set する。
- time-set(busy-dial(A,B):30) : busydial を 30 秒間 set する。
- time-set(busy(A):30) : busy を 30 秒間 set する。
- time-reset(dial-tone(A)) : dialtone を reset する。
- time-reset(busy-dial(A,B)) : busydial を reset する。
- time-reset(busy(A)) : busy を reset する。

(注) TASK の terminal は、端末（電話機）へ信号送出する。

以上のようなもので記述されているのが SDL 記述である。

4 UNIX 上の疑似実行システム

前節で概要を述べた制御プロセスに対して、入力を全て人間に指示してもらうことにより、信号送出動作、タスクと呼ばれる処理の実行、状態遷移を行うシステムを疑似実行システムと呼ぶ。

疑似実行システムの動作仕様を、以下のように設計した。

前提：

（1）自分の電話番号は、最初に人間が設定する。

（2）信号をやりとりする相手プロセスは、11,22,..,99（電話番号に対応）で特定されるもの以外は存在しない。

入力：

可能な入力をメニュー表示し、簡単に指定できるようにする。

出力：

信号出力動作の際には、出力する信号名、相手先をメッセージで表示する。

タスク：

実行される処理をメッセージで表示する。

実際の疑似システムの走行例を次に示す。

```
************************************
************** start ***************
************************************

 myself dial > 73

        ##### idle #####

************************************
*          ----- menu -----         *
*           select number           *
*       1, offhook                  *
*       6, s1                       *
*       7, s2                       *
************************************

 event input > 6
 signal > 22
 r1_s1:X = 22
        ##### idle #####

************************************
*          ----- menu -----         *
*           select number           *
*       1, offhook                  *
*       6, s1                       *
*       7, s2                       *
************************************
```

操作者は、まず自分の(電話)番号を入力する。疑似実行システムは、到達した状態と、その状態で定義されているevent(受信信号)を表示する。操作者は、システムが表示したeventの1つを番号で選択する。選択したeventが内部信号の場合には、通信相手の(電話)番号を更に指定する。システムは、入力されたeventに従って、タスクや出力処理があれば、これらを表示し、次に到達した状態と、そこで定義されたeventを表示する動作を繰り返す。

疑似実行システムでは、入力としては、自分の(電話)番号、event、任意の相手の(電話)番号の3つがあり、そして、出力としては、各STATEでのeventメニュー、TASK、OPT、STATE、電話をかける(発信の)場合の相手の(電話)番号のメニューがある。

eventメニューは、入力をわかりやすくするためと各STATEでの可能なeventを表示して次の動作をわかりやすくするためにある。

疑似実行システムの仕様が決まったので、次節では、SDL からこの疑似実行システムのためのCのプログラム記述を生成する手法について述べる。

5 UNIX 上の疑似実行システムで走行可能なCプログラムを生成するための条件

UNIX 上で走行可能なCプログラムを生成するためには、SDL のテキストをCプログラム変換ツール、KINDRA に通すときと KINDRA によって生成されたCプログラムを UNIX 上で実行可能なプログラムにするときにそれぞれ条件が必要になってくる。以下では、それぞれの条件について示すことにする。

KINDRA によって生成されるCプログラムを意識した時の SDL のテキストの条件

（1） event のリストの作成。

（2）「状態と event の形式変更」： SDL のテキストの STATE と IPT を EVT 化する。
(KINDRA のCへの変換部では、状態とその状態で定義された event を1つの EVT という名前の箱に入れた形式を前提としている。）例を以下に示す。

(変換前)

```
m1-m2,m47,m50
        STATE [idle(A)];
m2-m3
        IPT [offhook(A)];
m47-m48
        IPT [<s1:X];
m50-m51
        IPT [<s2:B];
```

(変換後)

```
m1-m2,m47,m50
        EVT [idle(A)
             !offhook(A)
             !s1:X
             !s2:B];
```

（3）「疑似実行システムの出力」： SDL のテキストの TASK と OPT を実行時に出力させるために printf 化する。

```
TASK [terminal(start:silent)]; を
TASK [printf(''terminal(start:silent)\n'')]; とする。
```

（4）「疑似実行システムの入力」： SDL のテキストの 内部信号の IPT を TASK に変える。
与えられた次の SDL の信号受信動作には、以下の２つの場合があり得る。

IPT [S 1 : B];

動作１） B が未定義であるときには、信号 S 1 受信時に、その信号を送った相手の番号を変数 B に代入し、記憶する。

動作２） B が既に値を持つ変数の場合には、信号 S 1 受信時に、その信号を送った相手の番号が、変数 B の値と同じであることを確認する。

そこで、この２つの異なる動作を実現するサブルーチンとして signal を用意し、これに合わせて、IPT 記述を以下のように変更する。

IPT [S 1 : B]; を TASK [B = signal (B)]; とする。

KINDRA によって生成されたＣプログラムを実行可能なプログラムにするための条件

（1） 標準ライブラリを付ける。

```
#include <stdio.h>
```

（2） event の define 設定。
今回は、event を数字に置き変えて数字で選択できるようにしたので event を define で設定しなければならない。以下に例を示す。

```
#define     offhook          1
#define     digit            2
#define     onhook           3
```

（3） 変数（X 、A 、B 、etc.）の宣言をする。
変数は、プログラム中で使われている文字をすべてピックアップして宣言（integer）しなければならない。以下に例を示す。

```
int  event , A , B , X , wrongno , nul ;
```

（4） メインプログラムの作成。
メインプログラムの例として以下に示すことにする。

```
main()
{
```

```
    printf(''\n\n'');
    printf(''\t************************************ \n'');
    printf(''\t*************** start *************** \n'');
    printf(''\t************************************ \n\n'');
    printf(''\t myself dial > '');
    scanf (''%d'',&A);
    printf(''\n\n'');
    pots();
    printf(''***************  end  *************** \n\n'');
}
```

（5）　各STATEでのeventのメニューの作成。

今回は各STATEでのeventのメニューは、以下のようにSTATEを表示してからeventメニュー表示するようにして、各STATEでのevent選択をわかりやすくした。

```
    printf(''\t\t##### idle ##### \n\n'');
    printf(''\t*************************************\n'');
    printf(''\t*          ----- menu -----          *\n'');
    printf(''\t*           select number           *\n'');
    printf(''\t*       1, offhook                  *\n'');
    printf(''\t*       6, s1                       *\n'');
    printf(''\t*       7, s2                       *\n'');
    printf(''\t*************************************\n\n'');
```

（6）　subroutineの作成。

今回、subroutineとしてはSDLのテキストで与えられたrequest(&event)のsubroutineと特定した相手の（電話）番号とキーボードから入力した（電話）番号が一致しているかどうかを判定するsubroutine、そして内部信号受信の際の受信相手の（電話）番号を操作者から受け取るsubroutineがある。以下にそれぞれのsubroutineを例に示す。

（6－1）　request(&event)の作成

SDLテキストのEVTボックスのCへの変換例を以下に示す。

（変換前）

```
m1-m2,m47,m50
        EVT [idle(A)
             !offhook(A)
             !s1:X
             !s2:B];
```

(変換後)

```
request(&event);
if ( event == offhook )
   対応する処理
else
   if ( event == s1 )
      対応する処理
else
   if ( event == s2 )
      対応する処理
```

上記の例からわかるように、event 受信はCプログラムの request 関数で処理される。この request 関数は外部から event を受信し、その信号名を変数 event に代入する処理を行なう。この request 関数の実行例を以下に示す。

```
request(event)
int  *event ;
{
   printf(''\t event input > '');
   scanf (''%d'',event);
}
```

(6－2) hantei()subroutine の作成

特定した相手の（電話）番号とキーボードから入力した（電話）番号が一致しているかどうかを判定する。

```
int hantei()
{
   int i ;
   for ( i=0 ; i<MAXGUEST ; i++ ){
      if ( X == guest[i] )
         break ;
   }
   if ( i >= MAXGUEST )
         X = wrongno ;
   return(X);
}
```

(6－3) signal(w)subroutine の作成

内部信号受信の際の受信相手の（電話）番号を操作者から受け取る。

```
int signal(w)
int  w ;
{
   int  AA ;
   printf(''\t signal > '');
   scanf (''%d'',&AA);
   if ( w == nul )
      w = AA ;
   else
      if ( !( w == nul ) )
         ;
   return(w);
}
```

SDL のテキストから UNIX 上で実行可能なＣプログラムにするためには、以上のようなプログラムが必要である。

6　おわりに

ATR 通信システム研究所では、通信ソフトウエア自動作成の研究を進めており、STR と呼ばれる動作記述から標準仕様記述言語 SDL への変換、及びこの SDL の詳細化手法を開発している。本実習では、この詳細化された SDL を出発点として、UNIX を目的システムとしたソフトウエア自動生成の再終段階の一部を担当した。

本実習では、UNIX システム上でのソフトウエア生成のために、SDL などのフローチャートを図的インタフェースで開発し、Ｃに変換する KINDRA システムを用いることを前提とした。また、実際にＣプログラムを実行では、１つの電話機を制御するプログラムを疑似的に実行するシステムの作成を目標とした。

SDL 記述のテキストから KINDRA によってＣプログラムを生成し、それを UNIX 上で走行可能なＣプログラムに変換した。具体的には、以下を行なった。

１） UNIX 上で動作する疑似実行システムの動作仕様を作成した。
２） Ｃへの変換のために必要な以下の条件を明らかにした。
　　i) 状態と event を KINDRA の定める EVT の形式に変換する。
３） 自動生成されたＣが、実際に実行可能なＣプログラムとして不足している部分を明らかにした。
　　i　）メインプログラムの欠如
　　ii ）各変数の宣言
　　iii) Ｃプログラムで用いられている入力受信サブルーチン request(event) の
　　サブルーチン本体

4）上記3）を補うプログラムを作成し、疑似実行システムの仕様に合わせ、システムの出力表示機能の付加を行ない、実際に、疑似実行システムプログラムを生成し、動作実験を行なった。

今回、UNIX上で実行可能なCプログラムを作成しましたが、SDLのテキストからKINDRAに通す時の条件付加やKINDRAによって生成されたCプログラムから実行可能なプログラムにする時の条件付加は、すべて手作業によって行なった。そのため、今後の課題としては、今回手作業で行なった条件付加を自動的に行なうプログラムを作成することである。

A SDL テキスト

```
VERSION    6  10-May-91 18:24:
        TITLE   [pots];
m0-m1
        TERM    [start];
m1-m2,m47,m50
        STATE   [idle(A)];
m2-m3
        IPT     [offhook(A)];
m3-m4
        TASK    [terminal
                !(start :dialtone)];
m4-m104
        TASK    [time-set(dial-
                !tone(A):20)];
m104-m5,m8,m38,m41,m44
        STATE   [dial-tone(A)];
m5-m6
        IPT     [onhook(A)];
m6-m7
        TASK    [time-reset
                !(dial-tone(A))];
m7-m107
        TASK    [terminal
                !(start :silent)];
m107-m99
        STATE   [idle(A)
                !<$>];
m8-m201
        IPT     [digit(A,X)];
m201-m202
        TASK    [terminal
                !(start :silent)];
m202-m203,m204
        TEST    [X == wrongno];
m204-m203,m9
        TEST    [X == A];
m9-m109
```

```
        TASK    [B:=X];
m109-m10
        OPT     [s2:B];
m10-m11,m25
        STATE   [wait];
m11-m12
        IPT     [<r2-s2:X];
m12-m13,m10
        TEST    [X == B];
m13-m14
        TASK    [time-reset
                !(dial-tone(A))];
m14-m114
        TASK    [terminal
                !(start :busytone)];
m114-m113
        TASK    [time-set
                !(busy-dial(A,B)
                !:30)];
m113-m15,m17,m19,m22
        STATE   [busy-dial(A,B)];
m15-m16
        IPT     [onhook(A)];
m16-m206
        TASK    [time-reset
                !(busy-dial(A,B))];
m206-m207
        TASK    [terminal
                !(start :silent)];
m207-m99
        STATE   [idle(A)
                !<$>];
m17-m18
        IPT     [timeover
                !(busy-dial(A,B))];
m18-m118
        TASK    [terminal
                !(start :howler)];
m118-m99
```

```
        STATE   [howler(A)
                !<$>];
m19-m20
        IPT     [<s1:X];
m20-m21
        OPT     [r1-s1:X];
m21-m99
        STATE   [busy-dial(A,B)
                !<$>];
m22-m23
        IPT     [<s2:C];
m23-m24
        OPT     [r2-s2:C];
m24-m99
        STATE   [busy-dial(A,B)
                !<$>];
m25-m26
        IPT     [<r1-s2:X];
m26-m27,m10
        TEST    [X == B];
m27-m28
        TASK    [time-reset(
                !dial-tone(A))];
m28-m128
        TASK    [terminal
                !(start :ringback)];
m128-m127
        TASK    [terminal
                !(hunt B)];
m127-m29,m32,m35
        STATE   [r-path(A,B),
                !ringback(A,B)];
m29-m30
        IPT     [onhook(A)];
m30-m31
        OPT     [sig-onhook:B];
m31-m131
        TASK    [terminal
                !(release B)];
```

```
m131-m132
        TASK    [terminal
                !(start :silent)];
m132-m99
        STATE   [idle(A)
                !<$>];
m32-m33
        IPT     [<s1:B];
m33-m34
        OPT     [r2-s1:B];
m34-m134
        TASK    [terminal
                !(connect B)];
m134-m135
        TASK    [terminal
                !(start :silent)];
m135-m99
        STATE   [path(A,B)
                !<$>];
m35-m36
        IPT     [<s2:C];
m36-m37
        OPT     [r2-s2:C];
m37-m99
        STATE   [r-path(A,B),
                !ringback(A,B)
                !<$>];
m203-m220
        TASK    [time-reset
                !(dial-tone(A))];
m220-m205
        TASK    [terminal
                !(start :busytone)];
m205-m301
        TASK    [time-set
                !(busy(A):30)];
m301-m99
        STATE   [busy(A)<$>];
m38-m39
```

```
        IPT     [timeover(
                !dial-tone(A))];
m39-m40
        TASK    [terminal
                !(start :busytone)];
m40-m140
        TASK    [time-set
                !(busy(A):30)];
m140-m99
        STATE   [busy(A)
                !<$>];
m41-m42
        IPT     [<s1:X];
m42-m43
        OPT     [r1-s1:X];
m43-m99
        STATE   [dial-tone(A)
                !<$>];
m44-m45
        IPT     [<s2:B];
m45-m46
        OPT     [r2-s2:B];
m46-m99
        STATE   [dial-tone(A)
                !<$>];
m47-m48
        IPT     [<s1:X];
m48-m49
        OPT     [r1-s1:X];
m49-m99
        STATE   [idle(A)
                !<$>];
m50-m51
        IPT     [<s2:B];
m51-m52
        OPT     [r1-s2:B];
m52-m152
        TASK    [terminal
                !(start :ringing)];
```

```
m152-m53,m91,m94,m97
        STATE   [ringing(A,B)];
m53-m54
        IPT     [offhook(A)];
m54-m55
        OPT     [s1:B];
m55-m56,m88
        STATE   [wait];
m56-m57
        IPT     [<r2-s1:X];
m57-m58,m55
        TEST    [X == B];
m58-m158
        TASK    [terminal
                !(start :silent)];
m158-m159
        TASK    [terminal
                !(connect B)];
m159-m59,m62,m65,m68
        STATE   [path(A,B)];
m59-m60
        IPT     [onhook(A)];
m60-m61
        OPT     [sig-onhook:B];
m61-m161
        TASK    [terminal
                !(release B)];
m161-m162
        TASK    [terminal
                !(start :silent)];
m162-m99
        STATE   [idle(A)
                !<$>];
m62-m63
        IPT     [<s1:X];
m63-m64
        OPT     [r1-s1:X];
m64-m99
        STATE   [path(A,B)
```

```
                !<$>];
m65-m66
        IPT     [<s2:C];
m66-m67
        OPT     [r2-s2:C];
m67-m99
        STATE   [path(A,B)
                !<$>];
m68-m69
        IPT     [sig-onhook:B];
m69-m169
        TASK    [terminal
                !(release B)];
m169-m170
        TASK    [terminal
                !(start :busytone)];
m170-m171
        TASK    [time-set
                !(busy(A):30)];
m171-m70,m72,m82,m85
        STATE   [busy(A)];
m70-m71
        IPT     [onhook(A)];
m71-m271
        TASK    [time-reset
                !(busy(A))];
m271-m272
        TASK    [terminal
                !(start :silent)];
m272-m99
        STATE   [idle(A)
                !<$>];
m72-m73
        IPT     [timeover(busy(A))];
m73-m174
        TASK    [terminal
                !(start :howler)];
m174-m74,m76,m79
        STATE   [howler(A)];
```

```
m74-m75
        IPT     [onhook(A)];
m75-m176
        TASK    [terminal
                !(start :silent)];
m176-m99
        STATE   [idle(A)
                !<$>];
m76-m77
        IPT     [<s1:X];
m77-m78
        OPT     [r1-s1:X];
m78-m99
        STATE   [howler(A)
                !<$>];
m79-m80
        IPT     [<s2:B];
m80-m81
        OPT     [r2-s2:B];
m81-m99
        STATE   [howler(A)
                !<$>];
m82-m83
        IPT     [<s1:X];
m83-m84
        OPT     [r1-s1:X];
m84-m99
        STATE   [busy(A)
                !<$>];
m85-m86
        IPT     [<s2:B];
m86-m87
        OPT     [r2-s2:B];
m87-m99
        STATE   [busy(A)
                !<$>];
m88-m89
        IPT     [<r1-s1:X];
m89-m90,m55
```

```
        TEST    [X == B];
m90-m99
        STATE   [ringing(A,B)
                !<$>];
m91-m92
        IPT     [<s1:X];
m92-m93
        OPT     [r1-s1:X];
m93-m99
        STATE   [ringing(A,B)
                !<$>];
m94-m95
        IPT     [<s2:C];
m95-m96
        OPT     [r2-s2:C];
m96-m99
        STATE   [ringing(A,B)
                !<$>];
m97-m98
        IPT     [sig-onhook:B];
m98-m198
        TASK    [terminal
                !(start :silent)];
m198-m99
        STATE   [idle(A)
                !<$>];
m99
        TERM    [exit];
```

B 変換後のSDLテキスト

```
VERSION    24  30-Jul-91 15:09:
TITLE   [pots];
m0-m1
        TERM    [start];
m1-m3,m47,m50
        EVT     [idle(A)
                !offhook(A)
                !s1:X
                !s2:X];
m3-m4
        TASK    :[printf("terminal(start :dialtone)\n"):];
m4-m104
        TASK    :[printf("time-set(dial-tone(A):20)\n"):];
m104-m6,m201,m39,m41,m44
        EVT     [dial-tone(A)
                !onhook(A)
                !digit(A,X)
                !timeover(dial-tone(A))
                !s1:X
                !s2:X];
m6-m7
        TASK    :[printf("time-reset(dial-tone(A))\n"):];
m7-m1
        TASK    :[printf("terminal(start :silent)\n"):];
m201-m202
        TASK    :[printf("terminal(start :silent)\n"):];
m202-m203,m204
        TEST    [X == wrongno];
m204-m203,m9
        TEST    [X == A];
m9-m109
        TASK    [B=X];
m109-m10
        OPT     :[printf("s2=%d\n",B):];
m10-m11,m26
        EVT     [wait
                !r2-s2:X
```

```
                !r1-s2:X];
m11-m12
        TASK    [X = signal(X)];
m12-m13,m10
        TEST    [X == B];
m13-m14
        TASK    :[printf("time-reset(dial-tone(A))\n"):];
m14-m114
        TASK    :[printf("terminal(start :busytone)\n"):];
m114-m113
        TASK    :[printf("time-set(busy-dial(A,B):30)\n"):];
m113-m16,m18,m19,m22
        EVT     [busy-dial(A,B)
                !onhook(A)
                !timeover(busy-dial(A,B))
                !s1:X
                !s2:X];
m16-m206
        TASK    :[printf("time-reset(busy-dial(A,B))\n"):];
m206-m1
        TASK    :[printf("terminal(start :silent)\n"):];
m18-m174
        TASK    :[printf("terminal(start :howler)\n"):];
m174-m75,m76,m79
        EVT     [howler(A)
                !onhook(A)
                !s1:X
                !s2:X];
m75-m1
        TASK    :[printf("terminal(start :silent)\n"):];
m76-m77
        TASK    [X = signal(X)];
m77-m174
        OPT     :[printf("r1-s1=%d\n",X):];
m79-m80
        TASK    [X = signal(X)];
m80-m174
        OPT     :[printf("r2-s2=%d\n",X):];
m19-m20
```

```
        TASK    [X = signal(X)];
m20-m113
        OPT     :[printf("r1-s1=%d\n",X):];
m22-m23
        TASK    [X = signal(X)];
m23-m113
        OPT     :[printf("r2-s2=%d\n",X):];
m26-m27,m10
        TEST    [X == B];
m27-m28
        TASK    :[printf("time-reset(dial-tone(A))\n"):];
m28-m128
        TASK    :[printf("terminal(start :ringback)\n"):];
m128-m127
        TASK    :[printf("terminal(hunt B)\n"):];
m127-m30,m32,m35
        EVT     [r-path(A,B),ringback(A,B)
                !onhook(A)
                !s1:B
                !s2:X];
m30-m31
        OPT     :[printf("sig-onhook=%d\n",B):];
m31-m131
        TASK    :[printf("terminal(release B)\n"):];
m131-m1
        TASK    :[printf("terminal(start :silent)\n"):];
m32-m33
        TASK    [B = signal(B)];
m33-m34
        OPT     :[printf("r2-s1=%d\n",B):];
m34-m134
        TASK    :[printf("terminal(connect B)\n"):];
m134-m159
        TASK    :[printf("terminal(start :silent)\n"):];
m159-m60,m62,m65,m68
        EVT     [path(A,B)
                !onhook(A)
                !s1:X
                !s2:C
```

```
                !sig-onhook:B];
m60-m61
        OPT     :[printf("sig-onhook=%d\n",B):];
m61-m161
        TASK    :[printf("terminal(release B)\n"):];
m161-m1
        TASK    :[printf("terminal(start :silent)\n"):];
m62-m63
        TASK    [X = signal(X)];
m63-m159
        OPT     :[printf("r1-s1=%d\n",X):];
m65-m66
        TASK    [X = signal(X)];
m66-m159
        OPT     :[printf("r2-s2=%d\n",X):];
m68-m69
        TASK    [B = signal(B)];
m69-m169
        TASK    :[printf("terminal(release B)\n"):];
m169-m170
        TASK    :[printf("terminal(start :busytone)\n"):];
m170-m171
        TASK    :[printf("time-set(busy(A):30)\n"):];
m171-m71,m73,m82,m85
        EVT     [busy(A)
                !onhook(A)
                !timeover(busy(A))
                !s1:X
                !s2:X];
m71-m271
        TASK    :[printf("time-reset(busy(A))\n"):];
m271-m1
        TASK    :[printf("terminal(start :silent)\n"):];
m73-m174
        TASK    :[printf("terminal(start :howler)\n"):];
m82-m83
        TASK    [X = signal(X)];
m83-m171
        OPT     :[printf("r1-s1=%d\n",X):];
```

```
m85-m86
        TASK    [X = signal(X)];
m86-m171
        OPT     :[printf("r2-s2=%d\n",X):];
m35-m36
        TASK    [X = signal(X)];
m36-m127
        OPT     :[printf("r2-s2=%d\n",X):];
m203-m220
        TASK    :[printf("time-reset(dial-tone(A))\n"):];
m220-m205
        TASK    :[printf("terminal(start :busytone)\n"):];
m205-m171
        TASK    :[printf("time-set(busy(A):30)\n"):];
m39-m40
        TASK    :[printf("terminal(start :busytone)\n"):];
m40-m171
        TASK    :[printf("time-set(busy(A):30)\n"):];
m41-m42
        TASK    [X = signal(X)];
m42-m104
        OPT     :[printf("r1-s1=%d\n",X):];
m44-m45
        TASK    [X = signal(X)];
m45-m104
        OPT     :[printf("r2-s2=%d\n",X):];
m47-m48
        TASK    [X = signal(X)];
m48-m1
        OPT     :[printf("r1-s1=%d\n",X):];
m50-m51
        TASK    [X = signal(X)];
m51-m52
        OPT     :[printf("r1-s2=%d\n",X):];
m52-m152
        TASK    :[printf("terminal(start :ringing)\n"):];
m152-m54,m91,m94,m97
        EVT     [ringing(A,B)
                !offhook(A)
```

```
                !s1:X
                !s2:X
                !sig-onhook:B];
m54-m55
        OPT     :[printf("s1=%d\n",B):];
m55-m56,m88
        EVT     [wait
                !r2-s1:X
                !r1-s1:X];
m56-m57
        TASK    [X = signal(X)];
m57-m58,m55
        TEST    [X == B];
m58-m158
        TASK    :[printf("terminal(start :silent)\n"):];
m158-m159
        TASK    :[printf("terminal(connect B)\n"):];
m88-m89
        TASK    [X = signal(X)];
m89-m152,m55
        TEST    [X == B];
m91-m92
        TASK    [X = signal(X)];
m92-m152
        OPT     :[printf("r1-s1=%d\n",X):];
m94-m95
        TASK    [X = signal(X)];
m95-m152
        OPT     :[printf("r2-s2=%d\n",X):];
m97-m98
        TASK    [B = signal(B)];
m98-m1
        TASK    :[printf("terminal(start :silent)\n"):];
;
```

C KINDRA によって生成されたCプログラム

```
/*  declaration of pots       chartname nere1  */
pots (  )
{
nere1m1:
   /*  idle(A)  */
   request(&event);
   if ( event == offhook(A) )
   {
      printf("terminal(start :dialtone)\n");
      printf("time-set(dial-tone(A):20)\n");
nere1m104:
      /*  dial-tone(A)  */
      request(&event);
      if ( event == onhook(A) )
      {
         printf("time-reset(dial-tone(A))\n");
         printf("terminal(start :silent)\n");
         goto nere1m1;
      }
      else
         if ( event == digit(A,X) )
         {
            printf("terminal(start :silent)\n");
            if ( !( X == wrongno ) )
            {
               if ( !( X == A ) )
               {
                  B=X;
                  printf("s2=%d\n",B);
nere1m10:
                  /*  wait  */
                  request(&event);
                  if ( event == r2-s2:X )
                  {
                     X = signal(X);
                     if ( !( X == B ) )
                        goto nere1m10;
```

```
            printf("time-reset(dial-tone(A))\n");
            printf("terminal(start :busytone)\n");
            printf("time-set(busy-dial(A,B):30)\n");
nere1m113:
            /*  busy-dial(A,B)  */
            request(&event);
            if ( event == onhook(A) )
            {
               printf("time-reset(busy-dial(A,B))\n");
               printf("terminal(start :silent)\n");
               goto nere1m1;
            }
            else
               if ( event == timeover(busy-dial(A,B)) )
                  printf("terminal(start :howler)\n");
               else
                  if ( event == s1:X )
                  {
                     X = signal(X);
                     printf("r1-s1=%d\n",X);
                     goto nere1m113;
                  }
                  else
                     if ( event == s2:X )
                     {
                        X = signal(X);
                        printf("r2-s2=%d\n",X);
                        goto nere1m113;
                     }
                     else
                        goto nere1m113;
         }
         else
            if ( event == r1-s2:X )
            {
               if ( !( X == B ) )
                  goto nere1m10;
               printf("time-reset(dial-tone(A))\n");
               printf("terminal(start :ringback)\n");
```

```
                    printf("terminal(hunt B)\n");
nere1m127:
                    /*  r-path(A,B),ringback(A,B)  */
                    request(&event);
                    if ( event == onhook(A) )
                    {
                       printf("sig-onhook=%d\n",B);
                       printf("terminal(release B)\n");
                       printf("terminal(start :silent)\n");
                       goto nere1m1;
                    }
                    else
                       if ( event == s1:B )
                       {
                          B = signal(B);
                          printf("r2-s1=%d\n",B);
                          printf("terminal(connect B)\n");
                          printf("terminal(start :silent)\n");
                       }
                       else
                          if ( event == s2:X )
                          {
                             X = signal(X);
                             printf("r2-s2=%d\n",X);
                             goto nere1m127;
                          }
                          else
                             goto nere1m127;
                    goto nere1m159;
                 }
                 else
                    goto nere1m10;
              goto nere1m174;
           }
        }
        printf("time-reset(dial-tone(A))\n");
        printf("terminal(start :busytone)\n");
        printf("time-set(busy(A):30)\n");
     }
```

```
        else
           if ( event == timeover(dial-tone(A)) )
           {
              printf("terminal(start :busytone)\n");
              printf("time-set(busy(A):30)\n");
           }
           else
              if ( event == s1:X )
              {
                 X = signal(X);
                 printf("r1-s1=%d\n",X);
                 goto nere1m104;
              }
              else
                 if ( event == s2:X )
                 {
                    X = signal(X);
                    printf("r2-s2=%d\n",X);
                    goto nere1m104;
                 }
                 else
                    goto nere1m104;
   }
   else
      if ( event == s1:X )
      {
         X = signal(X);
         printf("r1-s1=%d\n",X);
         goto nere1m1;
      }
      else
         if ( event == s2:X )
         {
            X = signal(X);
            printf("r1-s2=%d\n",X);
            printf("terminal(start :ringing)\n");
nere1m152:
            /*  ringing(A,B)  */
            request(&event);
```

```
            if ( event == offhook(A) )
            {
               printf("s1=%d\n",B);
nere1m55:
               /*  wait  */
               request(&event);
               if ( event == r2-s1:X )
               {
                  X = signal(X);
                  if ( !( X == B ) )
                     goto nere1m55;
                  printf("terminal(start :silent)\n");
                  printf("terminal(connect B)\n");
               }
               else
                  if ( event == r1-s1:X )
                  {
                     X = signal(X);
                     if ( X == B )
                        goto nere1m152;
                     goto nere1m55;
                  }
                  else
                     goto nere1m55;
            }
            else
               if ( event == s1:X )
               {
                  X = signal(X);
                  printf("r1-s1=%d\n",X);
                  goto nere1m152;
               }
               else
                  if ( event == s2:X )
                  {
                     X = signal(X);
                     printf("r2-s2=%d\n",X);
                     goto nere1m152;
                  }
```

```
            else
               if ( event == sig-onhook:B )
               {
                  B = signal(B);
                  printf("terminal(start :silent)\n");
                  goto nere1m1;
               }
               else
                  goto nere1m152;
nere1m159:
         /*  path(A,B)  */
         request(&event);
         if ( event == onhook(A) )
         {
            printf("sig-onhook=%d\n",B);
            printf("terminal(release B)\n");
            printf("terminal(start :silent)\n");
            goto nere1m1;
         }
         else
            if ( event == s1:X )
            {
               X = signal(X);
               printf("r1-s1=%d\n",X);
               goto nere1m159;
            }
            else
               if ( event == s2:C )
               {
                  X = signal(X);
                  printf("r2-s2=%d\n",X);
                  goto nere1m159;
               }
               else
                  if ( event == sig-onhook:B )
                  {
                     B = signal(B);
                     printf("terminal(release B)\n");
                     printf("terminal(start :busytone)\n");
```

```
                        printf("time-set(busy(A):30)\n");
                    }
                    else
                        goto nere1m159;
            }
            else
                goto nere1m1;
nere1m171:
    /*  busy(A)  */
    request(&event);
    if ( event == onhook(A) )
    {
        printf("time-reset(busy(A))\n");
        printf("terminal(start :silent)\n");
        goto nere1m1;
    }
    else
        if ( event == timeover(busy(A)) )
            printf("terminal(start :howler)\n");
        else
            if ( event == s1:X )
            {
                X = signal(X);
                printf("r1-s1=%d\n",X);
                goto nere1m171;
            }
            else
                if ( event == s2:X )
                {
                    X = signal(X);
                    printf("r2-s2=%d\n",X);
                    goto nere1m171;
                }
                else
                    goto nere1m171;
nere1m174:
    /*  howler(A)  */
    request(&event);
    if ( event == onhook(A) )
```

```
   {
      printf("terminal(start :silent)\n");
      goto nere1m1;
   }
   else
      if ( event == s1:X )
      {
         X = signal(X);
         printf("r1-s1=%d\n",X);
      }
      else
         if ( event == s2:X )
         {
            X = signal(X);
            printf("r2-s2=%d\n",X);
         }
         else
            goto nere1m174;
   goto nere1m174;
} /*  end of pots  */
```

D　疑似実行プログラム

```
#include <stdio.h>

#define     offhook_A                1
#define     digit_A_X                2
#define     onhook_A                 3
#define     r2_s2_X                  4
#define     timeover_busy_dial_A_B   5
#define     s1_X                     6
#define     s2_X                     7
#define     r1_s2_X                  8
#define     timeover_dial_tone_A     9
#define     r2_s1_X                  10
#define     r1_s1_X                  11
#define     sig_onhook_B             12
#define     timeover_busy_A          13
#define     s1_B                     14
#define     MAXGUEST                 9

static int guest[MAXGUEST] = { 11 , 22 , 33 , 44 , 55 , 66 , 77 , 88 , 99 };
int  event , A , B , X , wrongno , nul ;
   nul = 1001 ;
   wrongno = 0 ;


main()
{
   printf("\n\n");
   printf("\t************************************ \n");
   printf("\t*************** start *************** \n");
   printf("\t************************************ \n\n");
   printf("\t myself dial > ");
   scanf ("%d",&A);
   printf("\n\n");
   pots();
   printf("***************  end  *************** \n\n");
}
```

```
/*  declaration of pots      chartname nere  */
pots (  )
{
nerem1:
   X=nul;
   /*  idle(A)  */
   printf("\t\t##### idle ##### \n\n");
   printf("\t*************************************\n");
   printf("\t*          ----- menu -----          *\n");
   printf("\t*           select number           *\n");
   printf("\t*      1, offhook_A                 *\n");
   printf("\t*      6, s1_X                      *\n");
   printf("\t*      7, s2_X                      *\n");
   printf("\t*************************************\n\n");
   request(&event);
   if ( event == offhook_A )
   {
      printf("\t terminal(start :dialtone)\n");
      printf("\t time-set(dial-tone(A):20)\n");
nerem104:
      X=nul;
      /*  dial-tone(A)  */
      printf("\t\t##### dial_tone ##### \n");
      printf("\t*************************************\n");
      printf("\t*          ----- menu -----          *\n");
      printf("\t*           select number           *\n");
      printf("\t*      2, digit_A_B                 *\n");
      printf("\t*      3, onhook_X                  *\n");
      printf("\t*      6, s1_X                      *\n");
      printf("\t*      7, s2_X                      *\n");
      printf("\t*      9, timeover_dial_tone_A      *\n");
      printf("\t*************************************\n\n");
      request(&event);
      if ( event == onhook_A )
      {
         printf("\t time-reset(dial-tone(A))\n");
         printf("\t terminal(start :silent)\n");
         goto nerem1;
      }
```

```
        else
           if ( event == digit_A_X )
           {
              printf("\t*****************************************************\n");
              printf("\t*                                                   *\n");
              printf("\t*          ----- select dial number -----           *\n");
              printf("\t*                                                   *\n");
              printf("\t* { 11 , 22 , 33 , 44 , 55 , 66 , 77 , 88 , 99 } *\n");
              printf("\t*                                                   *\n");
              printf("\t*****************************************************\n\n");
              printf("\t dial > ");
              scanf ("%d",&X);
              printf("\t terminal(start :silent)\n");
              X = hantei(X,wrongno);
              if ( !( X == wrongno ) )
              {
                 if ( !( X == A ) )
                 {
                    B=X;
                    printf("s2=%d\n",B);
nerem10:
                    X=nul;
                    /*  wait  */
                    printf("\t\t##### wait_1 ##### \n");
                    printf("\t************************************\n");
                    printf("\t*          ----- menu -----         *\n");
                    printf("\t*           select number          *\n");
                    printf("\t*      4, r2_s2_X                  *\n");
                    printf("\t*      8, r1_s2_X                  *\n");
                    printf("\t************************************\n\n");
                    request(&event);
                    if ( event == r2_s2_X )
                    {
                       X = signal(X);
                       if ( !( X == B ) )
                          goto nerem10;
                       printf("\t time-reset(dial-tone(A))\n");
                       printf("\t terminal(start :busytone)\n");
                       printf("\t time-set(busy-dial(A,B):30)\n");
```

```
nerem113:
                X=nul;
                /*  busy-dial(A,B)  */
                printf("\t\t##### busy_dial ##### \n");
                printf("\t************************************\n");
                printf("\t*          ----- menu -----         *\n");
                printf("\t*           select number           *\n");
                printf("\t*      3, onhook_A                  *\n");
                printf("\t*      5, timeover_busy_dial_A_B    *\n");
                printf("\t*      6, s1_X                      *\n");
                printf("\t*      7, s2_X                      *\n");
                printf("\t************************************\n\n");
                request(&event);
                if ( event == onhook_A )
                {
                   printf("\t time-reset(busy-dial(A,B))\n");
                   printf("\t terminal(start :silent)\n");
                   goto nerem1;
                }
                else
                   if ( event == timeover_busy_dial_A_B )
                      printf("\t terminal(start :howler)\n");
                   else
                      if ( event == s1_X )
                      {
                         X = signal(X);
                         printf("r1-s1=%d\n",X);
                         goto nerem113;
                      }
                      else
                         if ( event == s2_X )
                         {
                            X = signal(X);
                            printf("r2-s2=%d\n",X);
                            goto nerem113;
                         }
                         else
                            goto nerem113;
             }
```

```
                    else
                       if ( event == r1_s2_X )
                       {
                          if ( !( X == B ) )
                             goto nerem10;
                          printf("\t time-reset(dial-tone(A))\n");
                          printf("\t terminal(start :ringback)\n");
                          printf("\t terminal(hunt B)\n");
nerem127:
                          X=nul;
                          /*  r-path(A,B),ringback(A,B)  */
                          printf("\t\t##### r_path(A,B),ringback(A,B) ##### \n");
                          printf("\t*************************************\n");
                          printf("\t*          ----- menu -----          *\n");
                          printf("\t*           select number           *\n");
                          printf("\t*       3, onhook_A                 *\n");
                          printf("\t*       6, s1_B                     *\n");
                          printf("\t*      14, s2_X                     *\n");
                          printf("\t*************************************\n\n");
                          request(&event);
                          if ( event == onhook_A )
                          {
                             printf("\t sig-onhook=%d\n",B);
                             printf("\t terminal(release B)\n");
                             printf("\t terminal(start :silent)\n");
                             goto nerem1;
                          }
                          else
                             if ( event == s1_B )
                             {
                                B = signal(B);
                                printf("\t r2-s1=%d\n",B);
                                printf("\t terminal(connect B)\n");
                                printf("\t terminal(start :silent)\n");
                             }
                             else
                                if ( event == s2_X )
                                {
                                   X = signal(X);
```

```
                        printf("\t r2-s2=%d\n",X);
                        goto nerem127;
                     }
                     else
                        goto nerem127;
                  goto nerem159;
               }
               else
                  goto nerem10;
            goto nerem174;
         }
      }
      printf("\t time-reset(dial-tone(A))\n");
      printf("\t terminal(start :busytone)\n");
      printf("\t time-set(busy(A):30)\n");
   }
   else
      if ( event == timeover_dial_tone_A )
      {
         printf("\t terminal(start :busytone)\n");
         printf("\t time-set(busy(A):30)\n");
      }
      else
         if ( event == s1_X )
         {
            X = signal(X);
            printf("\t r1-s1=%d\n",X);
            goto nerem104;
         }
         else
            if ( event == s2_X )
            {
               X = signal(X);
               printf("\t r2-s2=%d\n",X);
               goto nerem104;
            }
            else
               goto nerem104;
}
```

```
   else
      if ( event == s1_X )
      {
         X = signal(X);
         printf("\t r1-s1=%d\n",X);
         goto nerem1;
      }
      else
         if ( event == s2_X )
         {
            X = signal(X);
            B = X ;
            printf("\t r1-s2=%d\n",X);
            printf("\t terminal(start :ringing)\n");
nerem152:
            X=nul;
            /*  ringing(A,B)  */
            printf("\t\t##### ringing(A,B) ##### \n");
            printf("\t*************************************\n");
            printf("\t*          ----- menu -----          *\n");
            printf("\t*           select number           *\n");
            printf("\t*       1, offhook_A                *\n");
            printf("\t*       6, s1_X                     *\n");
            printf("\t*       7, s2_X                     *\n");
            printf("\t*      12, sig_onhook_B             *\n");
            printf("\t*************************************\n\n");
            request(&event);
            if ( event == offhook_A )
            {
               printf("\t s1=%d\n",B);
nerem55:
               X=nul;
               /*  wait  */
               printf("\t\t##### wait_2 ##### \n");
               printf("\t*************************************\n");
               printf("\t*          ----- menu -----          *\n");
               printf("\t*           select number           *\n");
               printf("\t*      10, r2_s1_X                  *\n");
               printf("\t*      11, r1_s1_X                  *\n");
```

```
    printf("\t************************************\n\n");
    request(&event);
    if ( event == r2_s1_X )
    {
       X = signal(X);
       if ( !( X == B ) )
          goto nerem55;
       printf("\t terminal(start :silent)\n");
       printf("\t terminal(connect B)\n");
    }
    else
       if ( event == r1_s1_X )
       {
          X = signal(X);
          if ( X == B )
             goto nerem152;
          goto nerem55;
       }
       else
          goto nerem55;
}
else
   if ( event == s1_X )
   {
      X = signal(X);
      printf("\t r1-s1=%d\n",X);
      goto nerem152;
   }
   else
      if ( event == s2_X )
      {
         X = signal(X);
         printf("\t r2-s2=%d\n",X);
         goto nerem152;
      }
      else
         if ( event == sig_onhook_B )
         {
            B = signal(B);
```

```
                printf("\t terminal(start :silent)\n");
                goto nerem1;
            }
            else
                goto nerem152;
nerem159:
        X=nul;
        /*  path(A,B)  */
        printf("\t\t##### path(A,B) ##### \n");
        printf("\t*************************************\n");
        printf("\t*          ----- menu -----          *\n");
        printf("\t*           select number           *\n");
        printf("\t*       3, onhook_A                 *\n");
        printf("\t*       6, s1_X                     *\n");
        printf("\t*       7, s2_X                     *\n");
        printf("\t*      12, sig_onhook_B             *\n");
        printf("\t*************************************\n\n");
        request(&event);
        if ( event == onhook_A )
        {
           printf("\t sig-onhook=%d\n",B);
           printf("\t terminal(release B)\n");
           printf("\t terminal(start :silent)\n");
           goto nerem1;
        }
        else
           if ( event == s1_X )
           {
              X = signal(X);
              printf("\t r1-s1=%d\n",X);
              goto nerem159;
           }
           else
              if ( event == s2_X )
              {
                 X = signal(X);
                 printf("\t r2-s2=%d\n",X);
                 goto nerem159;
              }
```

```
                else
                    if ( event == sig_onhook_B )
                    {
                        B = signal(B);
                        printf("\t terminal(release B)\n");
                        printf("\t terminal(start :busytone)\n");
                        printf("\t time-set(busy(A):30)\n");
                    }
                    else
                        goto nerem159;
            }
            else
                goto nerem1;
nerem171:
    X=nul;
    /*  busy(A)  */
    printf("\t\t##### busy ##### \n");
    printf("\t************************************\n");
    printf("\t*          ----- menu -----         *\n");
    printf("\t*           select number          *\n");
    printf("\t*      3, onhook_A                 *\n");
    printf("\t*      6, s1_X                     *\n");
    printf("\t*      7, s2_X                     *\n");
    printf("\t*     13, timeover_busy_A          *\n");
    printf("\t************************************\n\n");
    request(&event);
    if ( event == onhook_A )
    {
        printf("\t time-reset(busy(A))\n");
        printf("\t terminal(start :silent)\n");
        goto nerem1;
    }
    else
        if ( event == timeover_busy_A )
            printf("\t terminal(start :howler)\n");
        else
            if ( event == s1_X )
            {
                X = signal(X);
```

```
            printf("\t r1-s1=%d\n",X);
            goto nerem171;
         }
         else
            if ( event == s2_X )
            {
               X = signal(X);
               printf("\t r2-s2=%d\n",X);
               goto nerem171;
            }
            else
               goto nerem171;
nerem174:
   X=nul;
   /*  howler(A)  */
   printf("\t\t##### howler ##### \n");
   printf("\t*************************************\n");
   printf("\t*          ----- menu -----          *\n");
   printf("\t*           select number           *\n");
   printf("\t*      3, onhook_A                  *\n");
   printf("\t*      6, s1_X                      *\n");
   printf("\t*      7, s2_X                      *\n");
   printf("\t*************************************\n\n");
   request(&event);
   if ( event == onhook_A )
   {
      printf("\t terminal(start :silent)\n");
      goto nerem1;
   }
   else
      if ( event == s1_X )
      {
         X = signal(X);
         printf("\t r1-s1=%d\n",X);
      }
      else
         if ( event == s2_X )
         {
            X = signal(X);
```

```
            printf("\t r2-s2=%d\n",X);
         }
         else
            goto nerem174;
   goto nerem174;
} /*  end of pots  */


request(event)
int  *event ;
{
   printf("\t event input > ");
   scanf ("%d",event);
}


int hantei()
{
  int i ;
  for ( i=0 ; i<MAXGUEST ; i++ ){
     if ( X == guest[i] )
        break ;
  }
  if ( i >= MAXGUEST )
     X = wrongno ;
  return(X);
}


int signal(w)
int  w ;
{
   int  AA ;
   printf("\t signal > ");
   scanf ("%d",&AA);
   if ( w == nul )
      w = AA ;
   else
      if ( !( w == nul ) )
```

```
        ;
    return(w);
}
```

E 実行例

(電話の発信)

```
************************************
************** start ***************
************************************

 myself dial > 73


        ##### idle #####

************************************
*         ----- menu -----          *
*           select number           *
*      1, offhook                   *
*      6, s1                        *
*      7, s2                        *
************************************

 event input > 1
 terminal(start:dialtone)
 time_set(dial_tone(A):20)
        ##### dial_tone #####
************************************
*         ----- menu -----          *
*           select number           *
*      2, digit                     *
*      3, onhook                    *
*      6, s1                        *
*      7, s2                        *
*      9, timeover_dial_tone        *
************************************

 event input > 2
************************************************
*                                              *
*         ----- select dial number -----       *
*                                              *
```

```
* { 11 , 22 , 33 , 44 , 55 , 66 , 77 , 88 , 99 } *
*                                                 *
***************************************************

 dial > 11
 terminal(start:silent)
 s2:B = 11
        ##### wait_1 #####
************************************
*          ----- menu -----         *
*           select number           *
*       4, r2_s2                    *
*       8, r1_s2                    *
************************************

 event input > 8
 signal > 11
 time_reset(dial_tone(A))
 terminal(start:ringback)
 terminal(hunt_B)
        ##### r_path(A,B),ringback(A,B) #####
************************************
*          ----- menu -----         *
*           select number           *
*       3, onhookt                  *
*       6, s1                       *
*       7, s2                       *
************************************

 event input > 6
 signal > 11
 r2_s1:B = 11
 terminal(connect_B)
 terminal(start:silent)
        ##### path(A,B) #####
************************************
*          ----- menu -----         *
*           select number           *
*       3, onhook                   *
```

```
*      6, s1                        *
*      7, s2                        *
*     12, sig_onhook                *
**********************************

 event input > 3
 sig_onhook:B = 11
 terminal(release_B)
 terminal(start:silent)
        ##### idle #####

**********************************
*         ----- menu -----         *
*          select number           *
*      1, offhook                  *
*      6, s1                       *
*      7, s2                       *
**********************************
```

(電話の着信)

```
************************************
*************** start ***************
************************************

 myself dial > 73


        ##### idle #####

************************************
*          ----- menu -----         *
*           select number           *
*       1, offhook                  *
*       6, s1                       *
*       7, s2                       *
************************************

 event input > 7
 signal > 65
 r1_s2:X = 65
 terminal(start:ringing)
        ##### ringing(A,B) #####
************************************
*          ----- menu -----         *
*           select number           *
*       1, offhook                  *
*       6, s1                       *
*       7, s2                       *
*      12, sig_onhook               *
************************************

 event input > 1
 s1:B = 65
        ##### wait_2 #####
************************************
*          ----- menu -----         *
*           select number           *
*      10, r2_s1                    *
```

```
*     11, r1_s1                    *
**********************************

 event input > 10
 signal > 65
 terminal(start:silent)
 terminal(connect_B)
        ##### path(A,B) #####
**********************************
*          ----- menu -----        *
*           select number          *
*      3, onhook                   *
*      6, s1                       *
*      7, s2                       *
*     12, sig_onhook               *
**********************************

 event input > 3
 sig_onhook:B = 65
 terminal(release_B)
 terminal(start:silent)
        ##### idle #####

**********************************
*          ----- menu -----        *
*           select number          *
*      1, offhook                  *
*     -6, s1                       *
*      7, s2                       *
**********************************
```