



新人セミナー第9回

SSMの使い方

担当：岩井

目次

- 1.SSMとは
- 2.SSMのインストール
- 3.SSMを使う
- 4.ログを取る
- 5.ログを再生する
- 6.SSMwriteを作る上での注意点

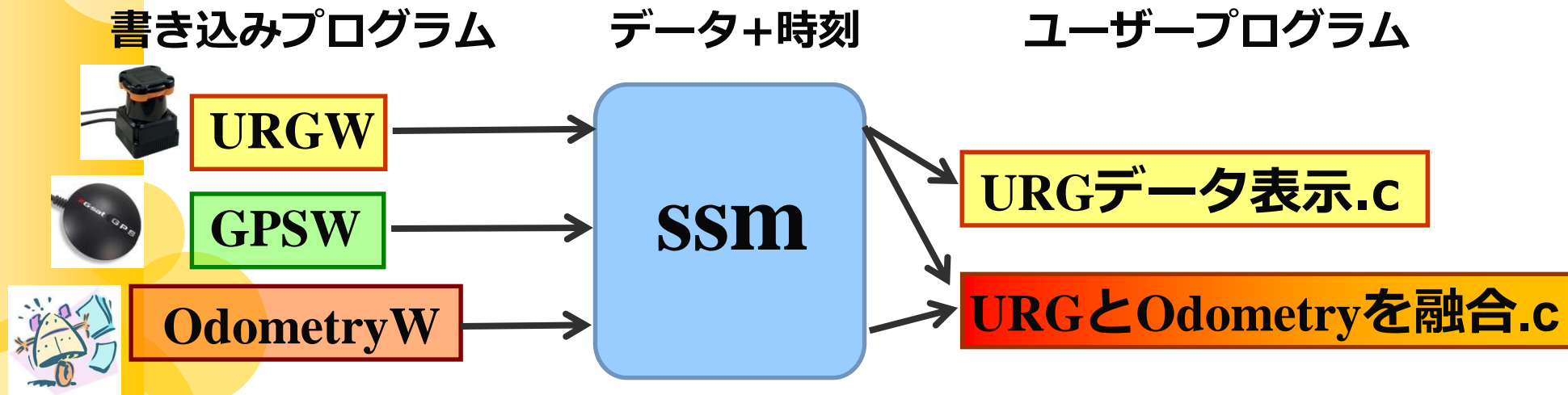


1. SSMとは

SSMとは

Sensor **S**haring **M**anager の略

センサデータを複数のプロセスで扱いやすくするためのプログラム



他の機能を考慮しなくてよくなるため容易にかつ
修正しやすいプログラムを作成出来る

SSMとは

特徴

構成がシンプル(write側、read側)
各センサにつき1入力n出力が可能
時間指定でのアクセスが可能
センサデータのログが取れる

制約

途中でデータサイズを変えられない
read側はデータ型を予め知っておく必要有
~~ログデータ再生時、一時停止・巻き戻しは出来ない~~



ssm-advance-player
により可能に！



2. SSMのインストール

SSMのインストール

- ① wikiからssm-0.8.0.tar.bz2をダウンロード(Wiki Topページ→SSM)
- ② `tar xjvf ssm-0.8.0.tar.bz2`
- ③ `cd ssm-0.8.0`
- ④ `./configure`
- ⑤ `make`
- ⑥ `sudo make install`
- ⑦ `sudo ld config`

ssm-coordinatorと入力してインストールされているか確認

```
iwai@LT10-T04: ~  
ファイル(E) 編集(E) 表示(V) 端末(T) ヘルプ(H)  
iwai@LT10-T04:~$ ssm-coordinator  
  
-----  
SSM ( Streaming data Sharing Manager )  
Ver. 0.8.0  
-----  
  
Message queue ready.  
Msg queue ID = 0
```

←このような画面になればOK

※gcc、gpp、g++をインストールしておく必要有



3. SSMを使う

SSMを使う

ssmにはデータ書き込み側(write)と読み込み側(read)が必要
まずは読み書き用のヘッダファイルを生成

SensorA.h: ssmで扱う構造体を定義

```
typedef struct {  
    int a;  
}SensorA;
```

今回は整数をssmにひたすら送るプログラムを作るので
構造体にはint型の変数を

送りたいデータに応じて構造体の中身は変える

char moji[256];

double distance[100];

など

SSMを使う

Write側の作成

int2ssmW.c : ssmに数値を 1 秒ごとに書きこむプログラム

```
#include <stdio.h>
#include <ssm.h>
#include "SensorA.h" //読み書き用のヘッダファイル

int main(void) {
    SensorA data;
    double measured_time;
    SSM_sid ssm_sid;

    //ssmの初期化
    initSSM();

    //名前をsensor_Aで、IDを0で、1秒おきに来るデータを3秒間保持するよう領域を確保
    ssm_sid=createSSM_time("Sensor_A",0,sizeof(SensorA),3,1);

    //初期値をSSMに書き込む
    data.a=0;
    writeSSM(ssm_sid, (char*) &data, gettimeSSM());

    //一秒ごとにデータを書き込み
    while (1) {
        data.a++;
        printf("%d\n",data.a);

        //SSMへのデータの書き込み。
        writeSSM(ssm_sid, (char*) &data, gettimeSSM());

        //一秒止める。ssmを使用するプログラムでは、sleepSSM(), usleepSSM()を使用することを推奨
        sleepSSM(1);
    }
    return 0;
}
```

※コンパイル時には「-lssm」を付ける
\$ gcc -o int2ssmW int2ssmW.c -lssm

SSMを使う

Read側の作成

int2ssmR.c : int2ssmWで書き込まれたデータを読み込むプログラム

```
#include <stdio.h>
#include <ssm.h>
#include "SensorA.h"

int main(void) {
    int tid;
    SensorA data; //sensor_Aのデータ取得用
    SSM_sid ssm_sid; //sensor_Aのアクセス用
    double measured_time; //sensor_Aの計測時刻取得用

    //初期化
    initSSM();

    //sensor_Aのオープン
    ssm_sid = openSSM("Sensor_A", 0, 0);

    //最新のデータのTIDを取得する
    tid=readSSM(ssm_sid, (char*)&data, &measured_time, -1);

    //1秒ごとにデータを取得
    while (1) {
        tid++;
        //最新のデータの読み込み 新しい入力がないなら0.5秒待つ
        while(readSSM(ssm_sid, (char*) &data, &measured_time, tid)<0) sleepSSM(0.5);
        printf("%d\n", data.a);
        sleepSSM(1);
    }
    return 0;
}
```

SSMを使う

SSMで用いる関数

たくさんあります

- `initSSM()`・・・SSMを利用するために必要な設定を初期化
- `createSSM_time(センサ名, ID, データサイズ、データ保持時間、記録する周期)`・・・新しいセンサを共有メモリ空間に登録
- `_gettimeSSM()`・・・現在時刻を取得
- `writeSSM(SID、書き込むデータのアドレス、計測時刻)`・・・センサデータの書き込み
- `sleepSSM(時間)`・・・指定された時間[s]だけプログラムを止める。ログを再生するときにデータが飛ばない
- `usleepSSM(時間)`・・・指定された時間[ms]だけプログラムを止める
- `openSSM(センサ名、ID、0)`・・・登録された共有メモリ上のセンサデータにアクセス
- `readSSM(SID、読み込むデータのアドレス、計測時刻を記録する変数のアドレス、時刻ID(-1で最新のもの))`・・・センサデータの読み出し

SID: データへのアクセスに必要なもの。SSM_sid型

深く考えすぎない

ssm.h

```
typedef char *SSM_sid; /* SIDは実はただのアドレス */
typedef int SSM_tid; /* TimeID型 */

/* ---- function prototypes ---- */
```

SSMを使う

SSMで用いる関数
最初は難しく感じますが

writeSSMでデータの書き込み

openSSMで利用するセンサを開く

readSSMでデータの読み込み

上の流れだけ覚えておけば使えるようになると思います

SSMを使う

プログラムを実行してみる

コンソールを3つ開き、下記のプログラムを順次起動する

①

\$ ssm-coordinator

②


\$ cd int2ssmWが存在する場所

\$./int2ssmW

③

\$ cd int2ssmRが存在する場所

\$./int2ssmR



```
iwai@LT10-T04: ~/ssmtestprog/in
ファイル(E) 編集(E) 表示(V) 端末(T) ヘルプ(H)
iwai@LT10-T04:~/ssmtestprog/intwrite$ ./int2ssmW
1
2
3
4
5
6
7
8
9
10
11
12
13

iwai@LT10-T04: ~/ssmtestprog/in
ファイル(E) 編集(E) 表示(V) 端末(T) ヘルプ(H)
iwai@LT10-T04:~/ssmtestprog/intread$ ./int2ssmR
2
3
4
5
6
7
8
9
10
11
12
13
14
```

書き込まれたデータが
read側で表示される

Write側に書き込まれたデータがRead側で次々読み込まれていることが確認出来ればOK

SSMをさらに使う

addR.c: int2ssmWで書き込まれたデータを足していくプログラム

```
#include <stdio.h>
#include <stdlib.h>
#include <ssm.h>
#include "SensorA.h"

int main(void) {
    int tid, sum=0;
    SensorA data; //sensor_Aのデータ取得用
    SSM_sid ssm_sid; //sensor_Aのアクセス用
    double measured_time; //sensor_Aの計測時刻取得用

    //初期化
    initSSM();

    //sensor_Aのオープン
    ssm_sid = openSSM("Sensor_A", 0, 0);

    //最新のデータのTIDを取得する
    tid=readSSM(ssm_sid, (char*)&data, &measured_time, -1);

    //1秒おきにデータを取得
    while (1) {
        tid++;
        //最新のデータの読み込み 新しい入力がないなら0.5秒待つ
        while(readSSM(ssm_sid, (char*)&data, &measured_time, tid)<0) sleepSSM(0.5);
        sum += data.a;
        printf("%d\n", sum);
        sleepSSM(1);
    }
    return 0;
}
```

ソースコードはwikiにあります

SSMをさらに使う

gnuR.c : int2ssmWの値 $\times \sin(x)$ のグラフを出力するプログラム

ソースコードはwikiにあります

```
#include <stdio.h>
#include <stdlib.h>
#include <ssm.h>
#include "SensorA.h"
#define GNUPLOT_PATH "/usr/bin/gnuplot"

int main(void) {
    int tid;
    SensorA data; //sensor_Aのデータ取得用
    SSM_sid ssm_sid; //sensor_Aのアクセス用
    double measured_time; //sensor_Aの計測時刻取得用
    FILE *gp=fopen(GNUPLOT_PATH, "w");

    //初期化
    initSSM();
    if(gp==NULL){
        fprintf(stderr, "error %s", GNUPLOT_PATH);
        exit(1);
    }

    //sensor_Aのオープン
    ssm_sid = openSSM("Sensor_A", 0, 0);

    //最新のデータのTIDを取得する
    tid=readSSM(ssm_sid, (char*)&data, &measured_time, -1);

    //gnuplotへの表示準備
    fprintf(gp, "set mouse\n");
    fprintf(gp, "set xlabel \"x\"\n");
    fprintf(gp, "set ylabel \"y\"\n");
    fprintf(gp, "set xrange [-10:10]\n");
    fprintf(gp, "set yrange [-10:10]\n");

    while(1)
    {
        tid++;
        //最新のデータの読み込み 新しい入力がないなら0.5秒待つ
        while(readSSM(ssm_sid, (char*)&data, &measured_time, tid)<0) sleepSSM(0.5);
        fprintf(gp, "plot %d*sin(x)\n", data.a);
        fflush(gp);
        printf("%d*sin(x)\n", data.a);
        sleepSSM(1);
    }
    return 0;
}
```


SSMをさらに使う

他のプログラムを実行してみる

①

```
$ ssm-coordinator
```

②

```
$ cd int2ssmWが存在する場所
```

```
$ ./int2ssmW
```

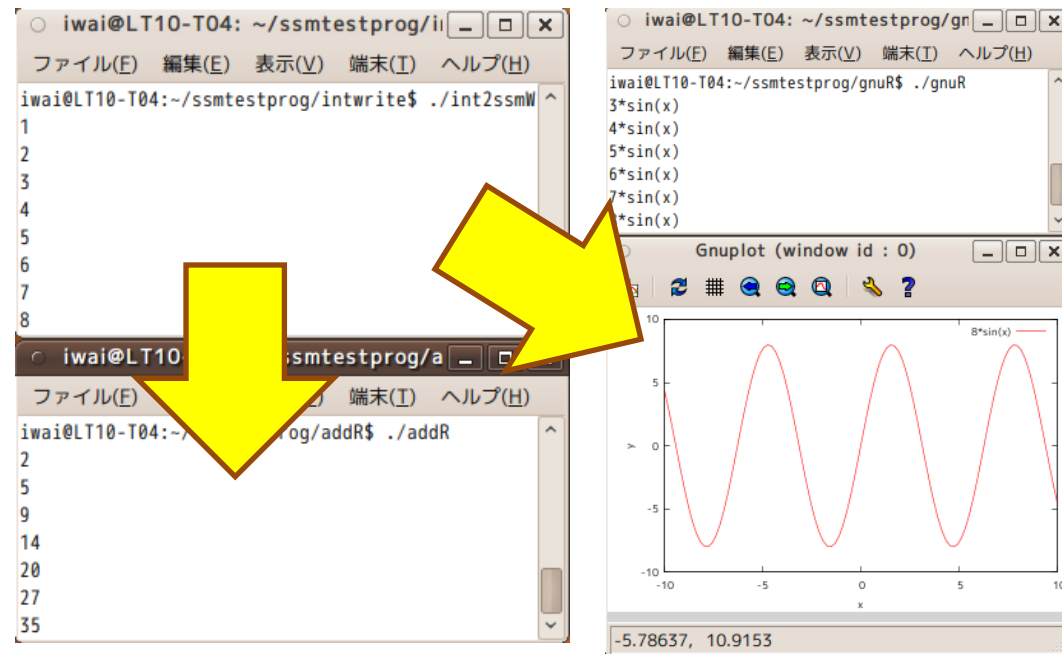
③

```
$ cd addRが存在する場所
```

```
$ ./addR
```

```
$ cd gnuRが存在する場所
```

```
$ ./gnuR
```



足し算

sin(x)のグラフ

write側を変えることなく様々なプログラムを実行することが可能

A decorative vertical bar on the left side of the slide, featuring a yellow-to-orange gradient and several overlapping circles of varying sizes and colors (yellow, orange, and light yellow).

4. ログを取る

ログを取る

書き込まれたデータのログを取る

ssmに書き込まれた順番に沿ってデータのログを取得

ログの取り方

\$ ssm-logger -n 記録したいデータ名 (-i センサのID) -l ログファイル名

①

\$ ssm-coordinator

②

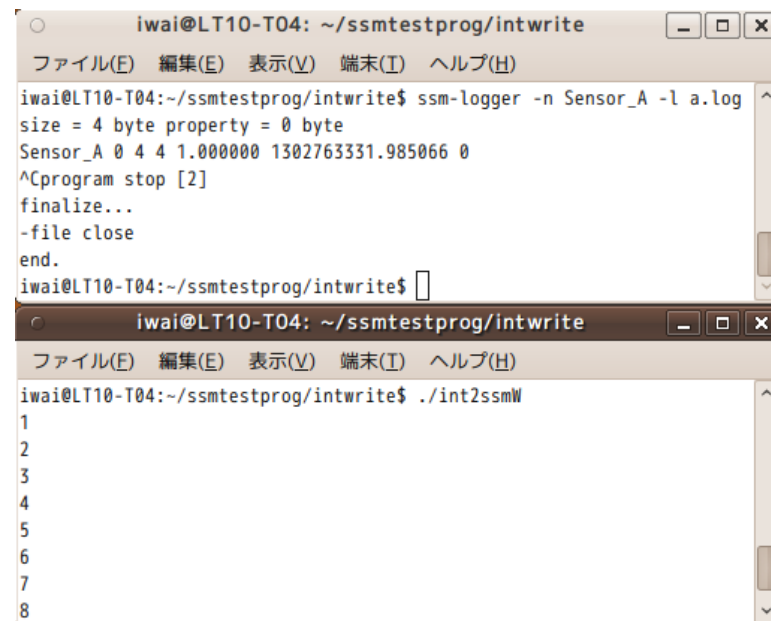
\$ cd int2ssmWが存在する場所

\$./int2ssmW

③

\$ cd ログを保存する場所

\$ ssm-logger -n Sensor_A -l a.log



The image shows two terminal windows. The top window shows the execution of `ssm-logger -n Sensor_A -l a.log`, which outputs sensor data and then terminates with `^Cprogram stop [2]`, `finalize...`, `-file close`, and `end.`. The bottom window shows the execution of `./int2ssmW`, which outputs a series of numbers (1 through 8) representing sensor data.

```
iwai@LT10-T04: ~/ssmtestprog/intwrite
ファイル(E) 編集(E) 表示(V) 端末(T) ヘルプ(H)

iwai@LT10-T04:~/ssmtestprog/intwrite$ ssm-logger -n Sensor_A -l a.log
size = 4 byte property = 0 byte
Sensor_A 0 0 4 1.000000 1302763331.985066 0
^Cprogram stop [2]
finalize...
-file close
end.
iwai@LT10-T04:~/ssmtestprog/intwrite$

iwai@LT10-T04: ~/ssmtestprog/intwrite
ファイル(E) 編集(E) 表示(V) 端末(T) ヘルプ(H)

iwai@LT10-T04:~/ssmtestprog/intwrite$ ./int2ssmW
1
2
3
4
5
6
7
8
```

適当なログが取れたらCtrl+cでログを停止

ログを取る

つくチャレで活躍

複数センサのログを取る

ターミナルを複数起動するか**シェルスクリプト**を使う

logger.sh

```
#!/bin/sh

if [ -d $1 ]; then
    echo "ERROR: Directory $1 already exists."
else
    mkdir $1
    ssm-logger -n Sensor_A -i 0 -l $1/Sensor_A.log &
    ssm-logger -n Sensor_B -i 0 -l $1/Sensor_B.log &
    ssm-logger -n Sensor_C -i 0 -l $1/Sensor_C.log &
fi
```

最後の行には
&を付けないように

起動するときは
./logger.sh ディレクトリ名

```
ファイル(E) 編集(E) 表示(V) 端末(T) ヘルプ(H)
iwai@LT10-T04:~/ssmtestprog$ ./logger.sh log
size = 4 byte property = 0 byte
Sensor_A 0 4 4 1.000000 1303816599.167558 0
size = 4 byte property = 0 byte
Sensor_B 0 4 4 1.000000 1303816599.167558 0
```

「log」ディレクトリに各種センサデータが
保存される

既に同名ディレクトリがあると怒られます



5. ログを再生する

ログを再生する

取ったログを再生してみる

ログの再生方法

\$ ssm-advance-player ログファイル名

①

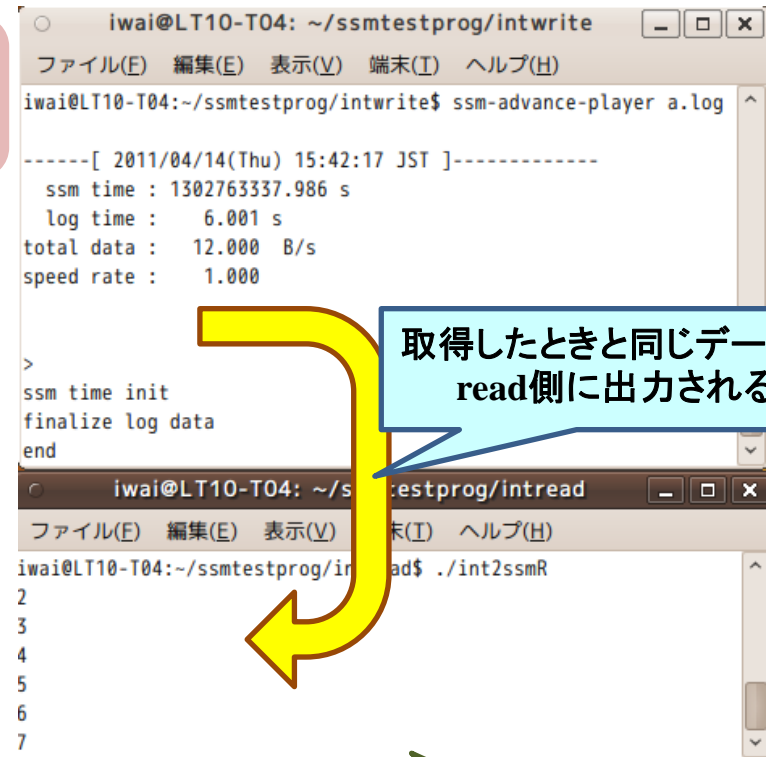
\$ cd ログを保存した場所

\$ ssm-advance-player a.log

②

\$ cd int2ssmRが存在する場所

\$./int2ssmR



```
iwai@LT10-T04: ~/ssmtestprog/intwrite
ファイル(E) 編集(E) 表示(V) 端末(T) ヘルプ(H)
iwai@LT10-T04:~/ssmtestprog/intwrite$ ssm-advance-player a.log
-----[ 2011/04/14(Thu) 15:42:17 JST ]-----
ssm time : 1302763337.986 s
log time : 6.001 s
total data : 12.000 B/s
speed rate : 1.000

>
ssm time init
finalize log data
end

iwai@LT10-T04: ~/ssmtestprog/intread
ファイル(E) 編集(E) 表示(V) 端末(T) ヘルプ(H)
iwai@LT10-T04:~/ssmtestprog/intread$ ./int2ssmR
2
3
4
5
6
7
```

取得したときと同じデータが
read側に出力される

同じデータで何度でもプログラムを試すことが出来る

同じ実験を繰り返す必要なし！

便利！

ログを再生する

再生時のコマンド

再生しているターミナルで以下のコマンドを打つことで一時停止などが可能に

- x** **数値** : 再生速度を変更
 - +** : 再生速度1.1倍(++++¥nなど連続入力可)
 - : 再生速度0.9倍(連続入力可)
- p** : 一時停止
- s** : ログ再生
- r** : 巻き戻し ←※readSSM_time()が使われていると失敗するとの記述有り
- q** : 終了

取得したいデータの所だけを再生することが可能！



6. SSMwriteを作る上 での注意点

SSMwriteを作る上での注意点

ssmに書き込むときの時間はデータを取得したときの時間にする

```
while(1){  
    GetData(&data);  
    sleep(1);  
    writeSSM(data_A,(char*)&data,_gettimeSSM());  
}
```

この時間だけデータ取得時刻と書き込み時刻がズれる

sleep以外にも**printfなどの描画関数**、**時間のかかる計算**が挿入されていると同じくズれます

対処法

データを取った時の時間を記録し、write時にそれを用いる

```
while(1){  
    time=gettimeSSM();  
    GetData(&data);  
    sleep(1);  
    writeSSM(data_A,(char*)&data, time);  
}
```

これでほぼ正確にデータを取得した時刻でssmに書くことが可能

課題

- ①int2ssmWの数字を受け取って5の倍数ならばOK、そうでない場合はNGを送るfive2ssmWの作成
- ②five2ssmWから送られた情報からターミナルにOKかNGを出力するfive2ssmRの作成

