TR-AC-0004

015

2次元プロットツール（FD付）

新上 和正　山田 順一
北川 美宏 (ATR-I)

1997. 3.31

ATR環境適応通信研究所

目次

1. はじめに

今回、２次元グラフにデータをプロットするツールを開発しました。
種々のシミュレーションや実験から得られる数値データを、簡易なフォーマットで指定するだけで２次元グラフにプロットできます。単機能ですが、数字の羅列からデータの傾向を把握するのにはとても使い勝手のよいツールです。グラフは、X-windowへ表示またはPS(Post Script)ファイルへ出力されますので、データの視覚化やドキュメント化にご活用ください。

2. 使用法

2.1. データファイル

本プロットツールを利用するためには、以下のフォーマットに準じたデータファイルを用意します。

```
sample graph      .....グラフ全体のタイトル

6   502   6       .....グラフ数, データ行数, データ列数

1  3  0          .....x軸のデータ列, y軸のデータ列, 点のみをプロット(=0)
2  3  1          .....x軸のデータ列, y軸のデータ列, 点を直線で連結(=1)
1  4  1
1  6  1
2  3  0
1  5  0

simulation time
generated packets
routed packets
lost packets
average packet delay
average call delay

100.0   547   362   339    67    66
200.0   598   544   499   100    94
300.0   434   521   545    92    98
400.0   468   437   494    91    94
500.0   375   433   385    92    80
600.0   387   421   458    75    80
700.0   504   439   441    77    81
800.0   486   500   478   106   104
```

グラフ数分(=6)の指定

各データ列のタイトル

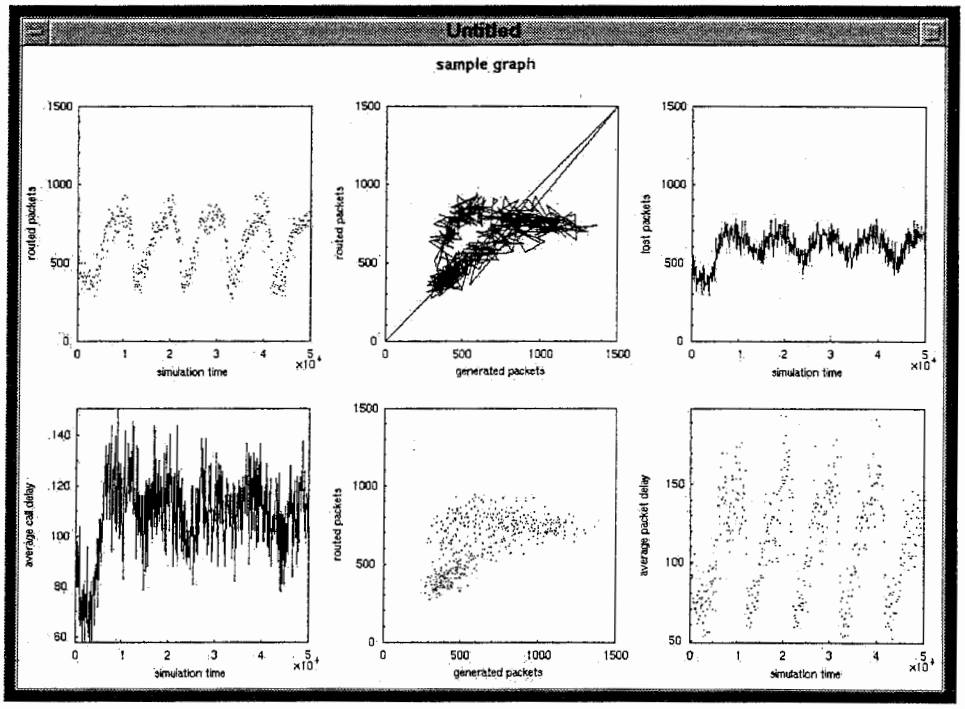
実データ（×502）

（注）データファイル中の空行は無視されます。

2.2. 実行

```
% plot -data data.1
```

この実行により、新しいウィンドウが開き次のようなグラフが表示されます。
開いたウィンドウは、（そのウィンドウにキーボードフォーカスがある状態で＝マウスカーソルをウィンドウ内に入れて）キーボードから'q'を入力すれば閉じられます。



実行時に以下のオプションが使用できます。

- -data <データファイル名>
  読み込むデータファイル名を指定します。無指定の場合、デフォルトとしてfort.99(Fortranのデフォルト出力ファイル)という名前のファイルが読み込まれます。

- -ps <PostScript ファイル名>
  出力するPostScriptファイル名を指定します。指定した場合、グラフはウィンドウ表示されません。

- -landscape
  横置きでウィンドウ表示またはPostScript出力されます(デフォルト)。

- -portrait
  縦置きでウィンドウ表示またはPostScript出力されます。

- -aspectratio <縦横比>
  グラフの縦横比を実数で指定します。1の場合、グラフが正方になります。

- -dotsize <点のサイズ>
  プロットする点の大きさをピクセル単位で指定します。無指定の場合、点のサイズは1です。

2.3. 稼働環境

HP-UX 10.xおよびSunOS 4.1xでの稼働は確認済みです。ウィンドウ表示では、X-window(X11R6)環境を前提としています。なお、X-window 環境があれば、他のプラットフォームでも問題なく稼働する可能性があります。

2.4. ファイル構成

本プロットツールは、以下のファイルで構成されています。

README
　プロットツール使用方法のドキュメント

bin/
　実行ファイルのディレクトリ

　plot_hp
　　HP用の実行ファイル

　plot_sun
　　SUN用の実行ファイル

sample/
　サンプルデータのディレクトリ

src/
　ソースプログラムのディレクトリ

　Imakefile
　　X環境上のMakefile作成パラメータファイル

　plot.c
　　プロットツールのメインモジュール

　draw.c
　　ウィンドウ表示／PostScript出力のモジュール

　graph.c
　　グラフ表示の汎用モジュール

　print.c
　　PostScript出力の汎用モジュール

　read.c
　　データファイル読み込みモジュール

　plot.h
　　プロットツールヘッダファイル

　graph.h
　　グラフ表示ヘッダファイル

2.5. 実行ファイルの作成

以下の手順により、各プラットフォーム用の実行ファイルを作成することができます。

1. 各ターゲットマシンへログインして、各プラットフォームの環境に合わせたMakefileを作成します。

```
% xmkmf
```

2. オブジェクトファイルをクリアします。

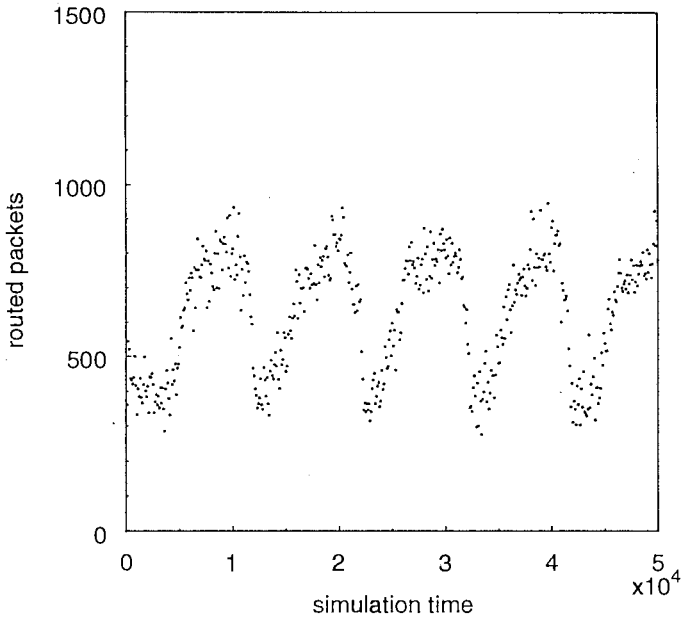
```
% make clean
```

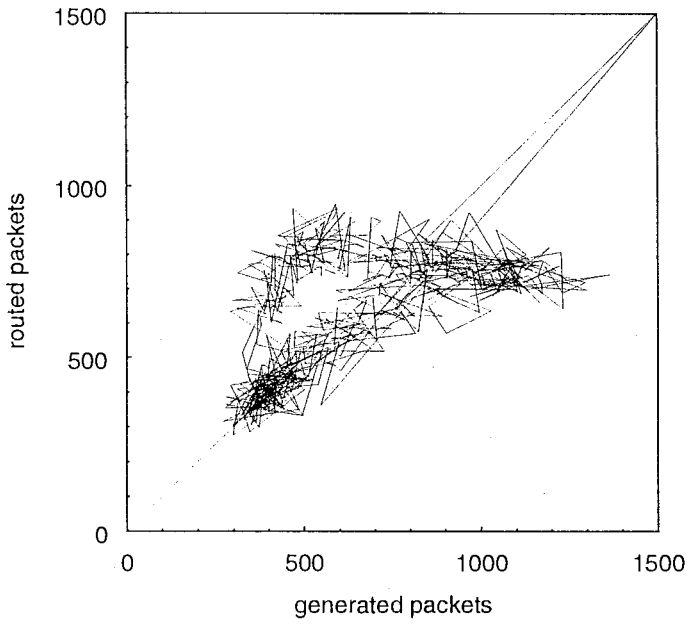
3. コンパイルして実行ファイルを作成します。

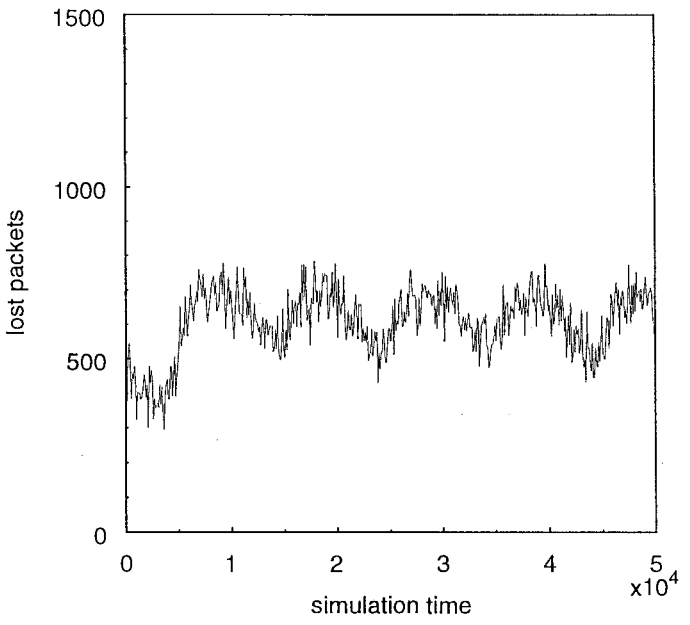
```
% make
```

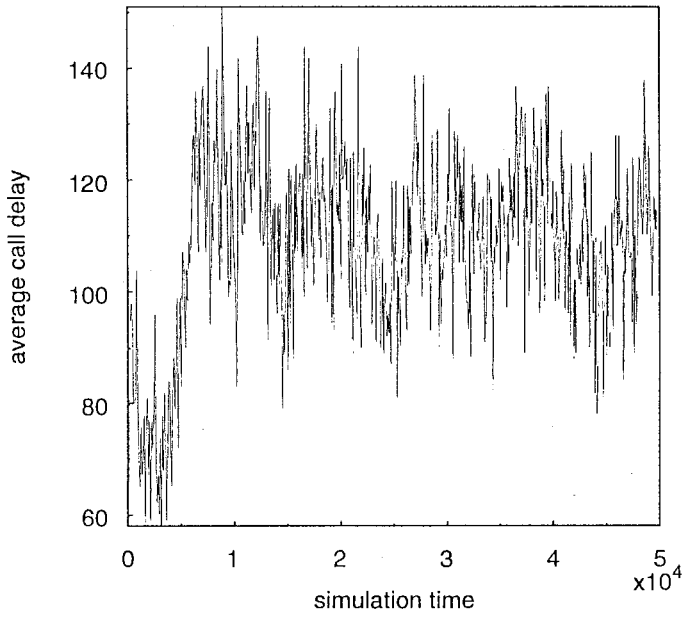
A.1 サンプルグラフおよびデータ

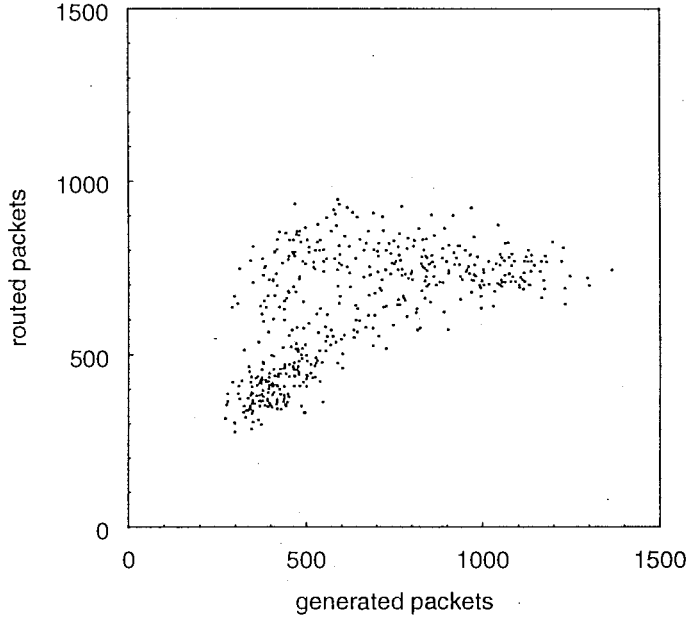
sample graph

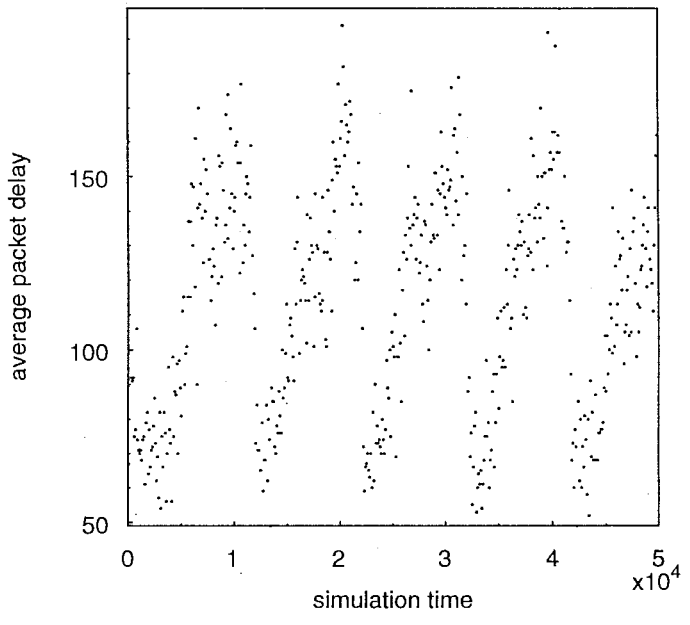












```
sample graph
6   502   6

1 3 0
2 3 1
1 4 1
1 6 1
2 3 0
1 5 0

simulation time
generated packets
routed packets
lost packets
average packet delay
average call delay
```

0.0	0	0	0	67	66
0.0	1500	1500	1500	67	66
100.0	547	362	339	67	66
200.0	598	544	499	100	94
300.0	434	521	545	92	98
400.0	468	437	494	91	94
500.0	375	433	385	92	80
600.0	387	421	458	75	80
700.0	504	439	441	77	81
800.0	486	500	478	106	104
900.0	368	417	442	74	80
1000.0	323	333	323	71	69
1100.0	438	408	405	70	65
1200.0	374	385	399	71	76
1300.0	452	437	401	68	67
1400.0	368	382	383	74	70
1500.0	377	366	387	75	78
1600.0	492	420	410	61	58
1700.0	402	499	454	79	77
1800.0	394	401	434	82	81
1900.0	404	384	387	64	67
2000.0	403	415	425	77	77
2100.0	336	338	299	66	59
2200.0	501	441	478	71	77
2300.0	466	439	429	72	72
2400.0	480	450	465	78	78
2500.0	292	420	443	86	96
2600.0	331	338	326	73	69
2700.0	418	370	384	62	62
2800.0	346	390	357	69	67
2900.0	447	361	370	57	60
3000.0	278	386	373	82	74
3100.0	401	352	360	54	58
3200.0	361	411	425	75	78
3300.0	390	366	367	66	66
3400.0	394	410	420	82	82
3500.0	325	350	341	76	75
3600.0	346	285	294	56	59
3700.0	501	408	394	70	69
3800.0	427	459	432	95	84
3900.0	318	424	439	77	84
4000.0	379	379	384	73	72
4100.0	494	332	384	56	65
4200.0	559	552	478	98	84
4300.0	343	420	472	75	89
4400.0	440	462	392	88	79
4500.0	501	430	455	87	87
4600.0	378	479	507	96	99
4700.0	392	390	391	70	72
4800.0	478	466	447	97	89
4900.0	545	478	491	89	99
5000.0	689	526	537	81	83
5100.0	559	614	656	111	107
5200.0	746	584	536	115	107
5300.0	507	589	555	90	98
5400.0	718	635	592	99	90
5500.0	764	642	567	101	104
5600.0	714	715	683	115	98
5700.0	712	683	625	137	109
5800.0	595	666	565	115	105
5900.0	780	730	621	137	108
6000.0	624	693	644	148	116
6100.0	724	743	716	130	127
6200.0	682	730	665	147	128
6300.0	822	575	643	118	119
6400.0	848	759	609	161	136
6500.0	758	642	654	90	125
6600.0	864	752	668	141	107
6700.0	863	842	697	170	129
6800.0	959	752	666	142	118
6900.0	962	732	760	148	130
7000.0	796	731	735	138	135
7100.0	1127	767	716	125	137
7200.0	1071	821	681	155	121
7300.0	1137	785	747	140	107
7400.0	1073	779	672	152	111
7500.0	1227	807	706	145	129
7600.0	1232	643	672	126	144
7700.0	1053	778	606	126	94
7800.0	1165	661	626	114	103
7900.0	1060	769	661	121	116
8000.0	1028	744	683	129	109
8100.0	1041	704	717	124	127
8200.0	1053	695	731	107	123
8300.0	833	800	741	136	127
8400.0	892	864	684	138	140
8500.0	1130	700	722	119	109
8600.0	950	799	637	156	102
8700.0	1003	699	649	153	120
8800.0	974	838	657	121	105
8900.0	788	834	740	154	151
9000.0	830	752	752	146	140
9100.0	806	828	688	131	119
9200.0	763	847	779	136	119
9300.0	748	827	718	168	125
9400.0	785	804	587	132	99
9500.0	855	903	632	174	102
9600.0	599	757	694	141	101
9700.0	631	910	737	164	129
9800.0	622	734	658	145	118
9900.0	538	799	700	129	107
10000.0	593	813	660	140	104
10100.0	469	934	602	144	97
10200.0	482	724	557	159	83
10300.0	549	767	630	160	113
10400.0	549	738	684	154	142
10500.0	443	849	767	122	131
10600.0	578	917	699	154	124
10700.0	463	790	631	177	111
10800.0	424	635	633	125	110
10900.0	385	754	629	136	120
11000.0	471	679	597	130	112
11100.0	461	793	765	145	128
11200.0	481	716	708	150	137
11300.0	377	775	739	148	121

11400.0	343	770	619	134	131
11500.0	382	748	676	144	119
11600.0	449	681	626	159	114
11700.0	375	616	694	127	129
11800.0	379	595	607	116	134
11900.0	338	466	559	106	118
12000.0	421	444	582	73	126
12100.0	405	408	665	84	137
12200.0	349	388	620	71	146
12300.0	383	352	608	71	142
12400.0	374	365	602	65	109
12500.0	360	381	577	65	119
12600.0	421	436	594	79	114
12700.0	418	362	539	59	108
12800.0	341	349	583	68	109
12900.0	525	468	622	68	118
13000.0	358	432	595	84	136
13100.0	428	386	529	62	91
13200.0	411	439	614	80	122
13300.0	279	364	613	74	135
13400.0	497	332	567	49	101
13500.0	408	458	580	89	108
13600.0	488	489	596	85	115
13700.0	443	445	596	85	102
13800.0	526	435	595	72	116
13900.0	466	477	520	70	100
14000.0	538	512	526	78	96
14100.0	452	472	586	76	116
14200.0	460	570	529	88	101
14300.0	395	434	625	81	116
14400.0	466	409	525	76	100
14500.0	489	518	506	100	79
14600.0	588	476	497	86	96
14700.0	527	492	545	89	89
14800.0	570	570	647	99	113
14900.0	451	555	519	113	118
15000.0	466	457	501	92	86
15100.0	513	472	585	91	122
15200.0	644	565	526	107	94
15300.0	530	616	703	109	121
15400.0	689	575	577	104	111
15500.0	575	553	554	98	88
15600.0	676	612	619	91	120
15700.0	780	654	672	129	107
15800.0	811	619	603	113	123
15900.0	679	752	658	131	107
16000.0	789	719	669	144	117
16100.0	694	612	593	99	121
16200.0	799	739	663	114	121
16300.0	936	657	657	120	104
16400.0	868	699	695	123	121
16500.0	1102	699	592	121	99
16600.0	847	701	773	119	144
16700.0	995	632	669	114	119
16800.0	976	757	774	102	104
16900.0	920	758	680	139	125
17000.0	1080	755	769	114	142
17100.0	1061	728	612	126	120
17200.0	1132	699	626	138	108
17300.0	1364	742	646	130	117
17400.0	1118	709	539	128	101
17500.0	997	711	678	101	107
17600.0	1051	714	661	115	120
17700.0	1196	823	660	145	130
17800.0	1078	709	784	130	120
17900.0	994	666	740	129	109

18000.0	1222	767	663	116	118
18100.0	1112	714	697	113	106
18200.0	1081	788	606	111	117
18300.0	1037	761	700	114	124
18400.0	1122	780	650	144	116
18500.0	918	792	680	128	117
18600.0	903	733	749	103	108
18700.0	904	814	723	101	98
18800.0	705	820	749	128	104
18900.0	739	790	741	146	115
19000.0	702	737	744	134	133
19100.0	662	722	617	126	94
19200.0	679	728	624	111	119
19300.0	690	908	693	149	93
19400.0	716	897	678	160	131
19500.0	676	855	753	140	136
19600.0	571	855	650	155	113
19700.0	669	820	684	153	116
19800.0	651	831	777	151	112
19900.0	508	808	691	177	125
20000.0	481	843	561	153	102
20100.0	609	840	733	161	141
20200.0	425	853	657	166	120
20300.0	594	933	629	194	121
20400.0	583	905	662	182	127
20500.0	449	772	659	156	118
20600.0	521	765	744	171	124
20700.0	433	763	598	165	103
20800.0	411	796	554	160	97
20900.0	407	716	581	163	124
21000.0	312	746	641	172	107
21100.0	418	802	587	168	88
21200.0	418	697	583	142	125
21300.0	438	652	631	147	110
21400.0	382	653	565	120	101
21500.0	438	629	587	125	95
21600.0	350	706	645	145	124
21700.0	387	634	689	120	144
21800.0	406	711	630	154	96
21900.0	444	660	558	134	90
22000.0	392	667	660	142	121
22100.0	325	514	659	106	112
22200.0	345	435	659	72	126
22300.0	403	347	548	59	99
22400.0	349	367	590	66	119
22500.0	339	341	550	67	106
22600.0	367	365	576	70	119
22700.0	392	348	546	65	112
22800.0	373	364	597	63	123
22900.0	272	316	494	70	94
23000.0	429	342	515	60	102
23100.0	310	409	517	82	110
23200.0	518	433	588	62	107
23300.0	340	451	508	90	91
23400.0	377	365	535	73	112
23500.0	448	386	580	78	114
23600.0	405	416	552	73	103
23700.0	385	426	580	74	90
23800.0	355	357	431	72	92
23900.0	438	372	516	70	106
24000.0	464	460	469	80	89
24100.0	347	407	517	74	99
24200.0	539	422	519	70	102
24300.0	568	529	586	77	92
24400.0	501	498	557	97	96
24500.0	463	466	493	89	92

24600.0	439	472	489	86	98
24700.0	634	549	539	100	87
24800.0	492	575	590	75	120
24900.0	602	503	587	105	111
25000.0	640	598	551	101	100
25100.0	472	531	590	98	109
25200.0	474	577	658	110	120
25300.0	602	461	512	69	81
25400.0	726	637	676	98	104
25500.0	645	630	581	98	98
25600.0	813	630	611	123	90
25700.0	635	549	600	102	106
25800.0	814	570	662	85	105
25900.0	760	656	682	117	119
26000.0	840	723	702	128	107
26100.0	861	724	593	104	95
26200.0	976	757	599	133	93
26300.0	827	780	638	126	119
26400.0	842	778	606	138	101
26500.0	842	741	632	153	117
26600.0	734	819	693	135	106
26700.0	984	780	624	130	115
26800.0	840	832	732	175	123
26900.0	1164	767	762	136	124
27000.0	885	777	720	139	139
27100.0	933	758	682	144	131
27200.0	969	714	684	126	117
27300.0	1116	734	688	138	127
27400.0	1048	787	674	142	121
27500.0	1300	697	568	122	111
27600.0	1091	709	590	125	99
27700.0	1117	768	678	133	113
27800.0	963	784	595	113	139
27900.0	1114	687	637	108	104
28000.0	1043	873	720	137	111
28100.0	1128	799	696	136	106
28200.0	979	725	708	124	110
28300.0	1063	820	715	114	107
28400.0	1103	692	692	100	93
28500.0	1002	735	677	120	100
28600.0	819	863	702	131	128
28700.0	974	729	687	142	120
28800.0	838	783	690	133	117
28900.0	830	830	655	132	110
29000.0	858	766	684	151	111
29100.0	762	785	687	151	129
29200.0	639	776	600	133	90
29300.0	803	811	655	146	111
29400.0	800	715	588	123	94
29500.0	718	857	728	145	104
29600.0	759	809	625	163	102
29700.0	586	871	699	153	110
29800.0	464	829	654	148	112
29900.0	621	811	753	144	106
30000.0	595	787	679	141	121
30100.0	608	841	550	139	116
30200.0	469	794	740	152	133
30300.0	588	743	645	146	112
30400.0	472	845	711	147	111
30500.0	490	767	635	149	88
30600.0	419	830	690	176	109
30700.0	580	775	709	136	129
30800.0	410	793	673	162	113
30900.0	549	801	657	164	121
31000.0	508	793	695	143	128
31100.0	525	745	631	157	105

31200.0	498	760	717	139	124
31300.0	350	811	704	179	111
31400.0	451	783	667	168	120
31500.0	379	728	599	120	105
31600.0	415	732	597	150	126
31700.0	492	650	566	127	103
31800.0	307	645	613	130	98
31900.0	438	599	625	113	98
32000.0	371	638	584	145	93
32100.0	366	536	620	88	112
32200.0	477	509	581	92	88
32300.0	276	354	613	69	110
32400.0	396	358	636	76	123
32500.0	418	343	594	55	103
32600.0	409	412	590	66	121
32700.0	423	445	596	78	114
32800.0	308	387	522	82	110
32900.0	373	298	532	53	109
33000.0	386	425	585	60	116
33100.0	299	301	521	64	104
33200.0	474	457	619	61	104
33300.0	313	372	611	65	113
33400.0	299	276	478	54	117
33500.0	451	378	540	61	91
33600.0	425	400	599	68	93
33700.0	480	481	605	75	97
33800.0	477	516	624	97	121
33900.0	390	371	603	75	120
34000.0	375	427	633	79	120
34100.0	443	350	540	60	105
34200.0	378	396	532	68	117
34300.0	435	460	473	71	82
34400.0	428	445	489	88	108
34500.0	396	446	527	93	105
34600.0	438	382	543	66	112
34700.0	513	447	543	79	108
34800.0	534	563	564	93	112
34900.0	497	529	598	109	122
35000.0	503	482	552	83	102
35100.0	532	483	617	98	120
35200.0	668	578	622	111	116
35300.0	607	556	625	95	119
35400.0	584	536	555	97	110
35500.0	593	646	526	112	108
35600.0	705	553	569	95	97
35700.0	694	675	717	123	114
35800.0	785	609	570	113	103
35900.0	589	696	649	130	124
36000.0	836	711	665	146	114
36100.0	1029	637	667	106	119
36200.0	902	571	567	109	109
36300.0	852	649	604	85	121
36400.0	1016	767	657	126	112
36500.0	836	660	638	121	137
36600.0	841	695	649	129	133
36700.0	827	682	617	123	108
36800.0	942	715	713	130	122
36900.0	903	741	665	139	126
37000.0	936	724	721	134	133
37100.0	1073	730	714	130	133
37200.0	1033	680	640	137	126
37300.0	1074	733	672	110	89
37400.0	1048	756	630	131	132
37500.0	1062	707	642	112	112
37600.0	967	678	692	109	123
37700.0	1046	700	568	127	99

37800.0	1088	768	682	129	105
37900.0	1145	756	650	139	123
38000.0	930	812	624	128	112
38100.0	967	923	674	161	119
38200.0	1083	723	721	123	133
38300.0	916	901	754	156	127
38400.0	825	760	756	138	102
38500.0	878	708	664	141	126
38600.0	726	799	677	131	125
38700.0	749	760	652	150	96
38800.0	965	787	596	142	100
38900.0	742	762	735	138	131
39000.0	770	927	745	170	117
39100.0	855	784	699	150	120
39200.0	704	782	658	132	98
39300.0	701	796	647	151	121
39400.0	663	800	625	151	136
39500.0	593	762	645	124	112
39600.0	615	752	779	141	137
39700.0	589	947	709	192	122
39800.0	526	799	713	152	113
39900.0	557	775	638	157	111
40000.0	533	875	659	152	120
40100.0	558	895	610	163	105
40200.0	521	750	566	155	98
40300.0	503	829	689	163	118
40400.0	474	854	600	188	115
40500.0	482	855	621	157	98
40600.0	498	866	721	162	105
40700.0	485	823	648	157	112
40800.0	453	829	687	199	129
40900.0	407	602	598	137	109
41000.0	395	715	705	135	123
41100.0	464	694	610	135	96
41200.0	297	666	687	150	111
41300.0	411	668	646	128	109
41400.0	451	677	580	131	96
41500.0	291	634	625	131	118
41600.0	457	523	505	114	92
41700.0	379	474	574	93	123
41800.0	348	357	631	68	116
41900.0	370	424	541	73	88
42000.0	365	312	664	60	103
42100.0	336	373	487	77	89
42200.0	369	350	507	72	108
42300.0	350	305	590	49	105
42400.0	373	417	520	85	102
42500.0	436	363	599	62	109
42600.0	363	437	571	80	101
42700.0	298	304	591	60	113
42800.0	419	409	515	72	105
42900.0	416	356	542	66	123
43000.0	375	457	645	81	115
43100.0	343	343	515	74	121
43200.0	350	337	474	58	97
43300.0	347	330	501	80	113
43400.0	486	350	433	52	90
43500.0	393	564	638	91	109
43600.0	441	354	530	69	125
43700.0	393	422	532	82	95
43800.0	409	391	492	68	109
43900.0	480	443	465	68	79
44000.0	408	496	548	87	110
44100.0	329	319	447	68	78
44200.0	418	412	447	68	84
44300.0	404	413	534	76	104

44400.0	371	366	491	76	108
44500.0	343	382	476	77	109
44600.0	482	410	549	77	100
44700.0	555	512	492	79	81
44800.0	527	476	548	89	103
44900.0	634	570	629	104	112
45000.0	555	539	514	88	88
45100.0	471	517	508	104	105
45200.0	726	518	499	93	92
45300.0	639	640	567	100	90
45400.0	677	669	622	113	98
45500.0	650	597	618	109	114
45600.0	554	579	532	99	93
45700.0	540	631	602	141	111
45800.0	891	621	690	101	116
45900.0	708	690	667	124	128
46000.0	773	667	596	107	108
46100.0	776	661	652	113	109
46200.0	887	706	694	134	128
46300.0	771	745	724	126	109
46400.0	1175	783	652	127	111
46500.0	1001	744	703	117	113
46600.0	1017	701	572	132	84
46700.0	994	691	682	96	108
46800.0	973	712	655	105	103
46900.0	1106	721	698	107	112
47000.0	990	698	688	117	122
47100.0	1135	739	619	121	106
47200.0	1173	687	644	104	106
47300.0	1171	713	604	129	107
47400.0	1161	738	659	105	92
47500.0	1176	782	776	146	124
47600.0	1245	725	674	134	89
47700.0	1179	766	604	138	113
47800.0	1231	689	717	119	94
47900.0	1295	719	631	98	100
48000.0	1137	767	721	113	119
48100.0	1054	707	638	105	124
48200.0	954	742	756	112	122
48300.0	855	765	678	123	114
48400.0	925	777	693	124	110
48500.0	897	767	701	130	114
48600.0	886	804	666	136	138
48700.0	952	737	707	144	127
48800.0	759	743	646	129	110
48900.0	770	727	656	118	112
49000.0	934	812	709	134	126
49100.0	835	746	741	141	117
49200.0	691	801	689	126	105
49300.0	753	768	656	123	99
49400.0	738	731	687	119	109
49500.0	766	770	695	111	117
49600.0	540	829	707	130	111
49700.0	616	924	689	141	115
49800.0	532	823	612	156	102
49900.0	644	896	571	162	100
50000.0	468	782	610	146	100

A2 プログラムソースリスト

```
/*
 * ACR PLOT TOOL Version 1.0
 *    (c) Copyright 1997
 *    ATR Adaptive Communications Research Laboratories
 *    All Rights Reserved
 */


/*
 * plot.c - プロットコマンド
 */

#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <stdio.h>

#include "plot.h"
#include "graph.h"

void main(int argc, char **argv)
{
  char filename[BUFSIZ+1];
  char *psfilename;             /* PSファイル名 */
  int  paperOrientation;        /* 紙の方向 */
  float aspectRatio;            /* 縦横比 */
  float dotSize;                /* 点の大きさ */
  int i;

  Display *display;
  Window mainWin;
  int screen;
  XSizeHints wmHint;
  int isDone;
  XEvent event;
  KeySym keySym;
  int key;
  char keyBuf[10];

  GC gc;

  DATA *data;

  /* コマンドライン引数の取得 */
  strcpy(filename, DEFAULT_FILENAME);
  psfilename = NULL;
  paperOrientation = DEFAULT_PAPER_ORIENTATION;
  aspectRatio = -1;
  dotSize = 1;
  for (i = 1;i < argc;i++) {
    if (!strcmp(argv[i], "-data")) {
      strcpy(filename, argv[++i]);
    }
    else if (!strcmp(argv[i], "-ps")) {
      psfilename = argv[++i];
    }
    else if (!strcmp(argv[i], "-landscape")) {
      paperOrientation = PAPER_LANDSCAPE;
    }
    else if (!strcmp(argv[i], "-portrait")) {
      paperOrientation = PAPER_PORTRAIT;
    }
    else if (!strcmp(argv[i], "-aspectratio")) {
      sscanf(argv[++i], "%g", &aspectRatio);
    }
    else if (!strcmp(argv[i], "-dotsize")) {
      sscanf(argv[++i], "%g", &dotSize);
    }
    else if (!strcmp(argv[i], "-help")) {
      usage();
      exit(0);
    }
    else {
      fprintf(stderr, "Unknown option '%s'.\n\n", argv[i]);
      usage();
      exit(1);
    }
  }

  /* データの読み込み */
  if ((data = ReadDataFile(filename)) == NULL)
    exit(1);

  data->aspectRatio = aspectRatio;
  data->orientation = paperOrientation;
  data->dotSize = dotSize;
  if (psfilename)
    data->mode = PS_ONLY_MODE;

  /* ディスプレイ接続 */
  if (data->mode == X_DRAW_MODE) {
    display = XOpenDisplay("");
    screen = DefaultScreen(display);
    data->display = display;
  }

  /* ウィンドウの生成 */
  CreatePlotWindows(data);

  if (data->mode == PS_ONLY_MODE) {
    SetPlotWindowSize(data);
    DrawGraph(data);
    PrintGraph(psfilename, data);
    exit(0);
  }

  isDone = 0;
  while (isDone == 0) {
    XNextEvent(display, &event);
    switch (event.type) {
    case Expose:
      if (event.xexpose.count == 0) {
#ifdef DEBUG
        fprintf(stderr, "OK Expose\n");
#endif
        RedrawGraph(data, event.xexpose.window);
      }
      break;

    case KeyPress:
      key = XLookupString(&event, keyBuf, 10, &key, NULL);
      if (key == 1 && keyBuf[0] == 'q')
        isDone = 1;
      else if (key == 1 && keyBuf[0] == 'p')
        PrintGraph("tmp.ps", data);
      break;

    case ConfigureNotify:
      if (event.xconfigure.window == data->mainWinId) {
#ifdef DEBUG
        fprintf(stderr, "OK ConfigureNotify(main)\n");
#endif
        /* ウィンドウの大きさの再設定 */
```

```
        data->mainWinWidth = event.xconfigure.width;
        data->mainWinHeight = event.xconfigure.height;
        SetPlotWindowSize(data);
        DrawGraph(data);
        if (psfilename) {
          PrintGraph(psfilename, data);
          isDone = 1;
        }
      }

      break;

    case DestroyNotify:
#ifdef DEBUG
      printf("OK Destroy\n");
#endif
      XDestroyWindow(data->display, data->mainWinId);
      XCloseDisplay(data->display);
      break;
    }
  }

  XDestroyWindow(data->display, data->mainWinId);
  XCloseDisplay(data->display);

  exit(0);
}

int usage()
{
  fprintf(stderr, "\nUsage: plot [options...]\n");
  fprintf(stderr, "  -data <file>  data filename. default is 'fort.99'.\n");
  fprintf(stderr, "  -ps   <file>  output PS filename.\n");
  fprintf(stderr, "  -landscape    landscape.\n");
  fprintf(stderr, "  -portrait     portrait.\n");
  fprintf(stderr, "  -aspectratio  aspect ratio.\n");
  fprintf(stderr, "  -dotsize      dot size by pixel. defaut is 1.\n");
  fprintf(stderr, "\n");
}
```

```
/*
 * ACR PLOT TOOL Version 1.0
 *     (c) Copyright 1997
 *     ATR Adaptive Communications Research Laboratories
 *     All Rights Reserved
 */


/*
 * draw.c - 描画関数
 */

#include <X11/X.h>
#include <X11/Xlib.h>
#include <X11/Xutil.h>

#include "graph.h"
#include "plot.h"

/* グラフハンドル配列 */
int gGraphNo[MAX_GRAPH_COUNT];

int DrawGraph(DATA *data)
{
  int i;
  int colx, coly;
  float ratio;
  float size;

  if (data->mode == PS_ONLY_MODE)
    GrSetPSOnlyMode();

  /* グラフの初期化 */
  GrInit();

  for (i = 0;i < data->nGraphs;i++) {
    colx = data->colX[i];
    coly = data->colY[i];

    /* グラフのオープン */
    gGraphNo[i] = GrOpen(data->display, data->winId[i]);

    /* 描画ウィンドウのサイズの設定 */
    GrSetWindowSize(gGraphNo[i],
                    data->winWidth[i], data->winHeight[i]);

    /* グラフのプロット */
    GrPlot(gGraphNo[i],
           data->table + (colx-1)*data->nRows,
           data->nValidRows,
           data->table + (coly-1)*data->nRows,
           data->nValidRows);

    /* グラフのプロパティの設定 */
    ratio = data->aspectRatio;
    size = data->dotSize;
    GrSet(gGraphNo[i],
          GR_MAX_EXP, 3,
          GR_XLABEL, data->colHeader[colx-1],
          GR_YLABEL, data->colHeader[coly-1],
          GR_BOX, True,
          GR_LINE_STYLE,
            (data->lineType[i] == 0 ? GR_LINE_STYLE_DOT:
             (data->lineType[i] == 1 ? GR_LINE_STYLE_LINE:
              (data->lineType[i] == 2 ? GR_LINE_STYLE_DOTTEDLINE:
               GR_LINE_STYLE_LINE))),
          GR_DOT_SIZE, &size,
          GR_DECORATION_COLOR, "",
          GR_GRAPH_TYPE, GR_TYPE_XY,
          GR_ASPECT_RATIO, &ratio,
          NULL);

/*    if (data->mode == X_DRAW_MODE)
      GrDraw(gGraphNo[i]);*/
  }

  /* タイトルウィンドウの描画 */
  if (data->mode == X_DRAW_MODE)
    grDrawTitleCenter(gGraphNo[0], data->titleWinId, data->title);

  if (data->mode == PS_ONLY_MODE)
    GrResetPSOnlyMode();

  return(0);
}

/*
 * グラフウィンドウの再描画
 */
int RedrawGraph(DATA *data, Window targetWin)
{
  int i;

  /* ターゲットのウィンドウのみを描画する */
  for (i = 0;i < data->nGraphs;i++) {
    if (GrGetWindow(gGraphNo[i]) == targetWin) {
      GrDraw(gGraphNo[i]);
      return(0);
    }
  }

  /* タイトルウィンドウの描画 */
  grDrawTitleCenter(gGraphNo[0], data->titleWinId, data->title);

  return(0);
}

int PrintGraph(char *filename, DATA *data)
{
  int i;
  FILE *fp;
  float x, y, width, height;
  float paperWidth, paperHeight;
  float topMargin, bottomMargin, leftMargin, rightMargin;
  float scaleFont;

  /* PS出力ファイルのオープン */
  if ((fp = fopen(filename, "w")) == NULL) {
    fprintf(stderr, "Cannot open file '%s'.\n", filename);
    return(-1);
  }

  /* 各ページパラメータの設定(A4,by inch) */
  paperWidth   = 8.5;
  paperHeight  = 11.7;
  topMargin    = 0.4;
  bottomMargin = 0.4;
  leftMargin   = 0.7;
  rightMargin  = 0.4;
  scaleFont    = 0.75;                    /* フォントの全体的なスケーリング */

  /* ヘッダ出力 */
```

```
  fprintf(fp, "%%!PS-Adobe-1.0\n");
  fprintf(fp, "%%%% DocumentFonts: Helvetica\n");
  fprintf(fp, "%%%% Title: %s\n", data->title);
  fprintf(fp, "%%%% Creator: \n");
  fprintf(fp, "%%%% CreationDate: \n");
  fprintf(fp, "%%%% For: \n");
  fprintf(fp, "%%%% Page: 1\n");
  fprintf(fp, "/inch { 72 mul } def\n");
  fprintf(fp, "/TopMargin    %g inch def\n", topMargin);
  fprintf(fp, "/BottomMargin %g inch def\n", bottomMargin);
  fprintf(fp, "/LeftMargin   %g inch def\n", leftMargin);
  fprintf(fp, "/RightMargin  %g inch def\n", rightMargin);
  fprintf(fp, "/PaperWidth   %g inch def\n", paperWidth);
  fprintf(fp, "/PaperHeight  %g inch def\n", paperHeight);
  fprintf(fp, "/TopLeftPos { LeftMargin PaperHeight TopMargin sub } def\n");
  fprintf(fp, "\n");
  fprintf(fp, "%% define font information\n");
  fprintf(fp, "/TheFontName /Helvetica def\n");
  fprintf(fp, "/TheFontSize  12 def\n");
  fprintf(fp, "/TheLineWidth 1 def                %% linewidth scaling factor\n");
  fprintf(fp, "/TheScaleX    1 def                %% x-axis scaling factor\n");
  fprintf(fp, "/TheScaleY    1 def                %% y-axis scaling factor\n");
  fprintf(fp, "/TheScaleFont 0.9 def              %% font scaling factor\n", scaleFont
);
  fprintf(fp, "\n");
  fprintf(fp, "%% define procedures\n");
  fprintf(fp, "/dot1 {\n");
  fprintf(fp, "%% called as: x y dot1\n");
  fprintf(fp, "TheScaleY mul /y exch def \n");
  fprintf(fp, "TheScaleX mul /x exch def\n");
  /* 指定された点の大きさの1/4を設定 */
  fprintf(fp, "x y %g 0 360 arc fill\n", 0.5 + (data->dotSize-1)/4);
  fprintf(fp, "} def\n");
  fprintf(fp, "\n");
  fprintf(fp, "/line { \n");
  fprintf(fp, "%% called as: x1 y1 x2 y2 line\n");
  fprintf(fp, "TheScaleY mul /y2 exch def\n");
  fprintf(fp, "TheScaleX mul /x2 exch def\n");
  fprintf(fp, "TheScaleY mul /y1 exch def\n");
  fprintf(fp, "TheScaleX mul /x1 exch def\n");
  fprintf(fp, "x1 y1 moveto\n");
  fprintf(fp, "x2 y2 lineto\n");
  fprintf(fp, "stroke\n");
  fprintf(fp, "} def\n");
  fprintf(fp, "\n");
  fprintf(fp, "/rectangle { \n");
  fprintf(fp, "%% called as: x1 y1 x2 y2 rectangle\n");
  fprintf(fp, "TheScaleY mul /y2 exch def\n");
  fprintf(fp, "TheScaleX mul /x2 exch def\n");
  fprintf(fp, "TheScaleY mul /y1 exch def\n");
  fprintf(fp, "TheScaleX mul /x1 exch def\n");
  fprintf(fp, "x1 y1 moveto\n");
  fprintf(fp, "x1 y2 lineto\n");
  fprintf(fp, "x2 y2 lineto\n");
  fprintf(fp, "x2 y1 lineto\n");
  fprintf(fp, "closepath\n");
  fprintf(fp, "stroke\n");
  fprintf(fp, "} def\n");
  fprintf(fp, "\n");
  fprintf(fp, "/centerText {\n");
  fprintf(fp, "%% called as: text x y w h centerText\n");
  fprintf(fp, "TheScaleY mul /height exch def\n");
  fprintf(fp, "TheScaleX mul /width exch def\n");
  fprintf(fp, "TheScaleY mul /y exch def\n");
  fprintf(fp, "TheScaleX mul /x exch def\n");
  fprintf(fp, "dup\n");
```

```
  fprintf(fp, "stringwidth\n");
  fprintf(fp, "pop\n");
  fprintf(fp, "/sh TheFontSize TheScaleFont mul def\n");
  fprintf(fp, "/sw exch def\n");
  fprintf(fp, "width sw sub 2 div x add\n");
  fprintf(fp, "height sh sub 2 div y add\n");
  fprintf(fp, "moveto\n");
  fprintf(fp, "show\n");
  fprintf(fp, "} def\n");
  fprintf(fp, "\n");
  fprintf(fp, "/centerTextRot90 {\n");
  fprintf(fp, "%% called as: text x y w h centerTextRot90\n");
  fprintf(fp, "TheScaleX mul /height exch def\n");
  fprintf(fp, "TheScaleY mul /width exch def\n");
  fprintf(fp, "TheScaleY mul /y exch def\n");
  fprintf(fp, "TheScaleX mul /x exch def\n");
  fprintf(fp, "dup\n");
  fprintf(fp, "stringwidth\n");
  fprintf(fp, "pop\n");
  fprintf(fp, "/sh TheFontSize TheScaleFont mul def\n");
  fprintf(fp, "/sw exch def\n");
  fprintf(fp, "height sh sub 2 div x add\n");
  fprintf(fp, "width sw sub 2 div y add\n");
  fprintf(fp, "moveto\n");
  fprintf(fp, "90 rotate\n");
  fprintf(fp, "show\n");
  fprintf(fp, "-90 rotate\n");
  fprintf(fp, "} def\n");
  fprintf(fp, "\n");
  fprintf(fp, "/rightText {\n");
  fprintf(fp, "%% called as: text x y rightText\n");
  fprintf(fp, "TheScaleY mul /y exch def\n");
  fprintf(fp, "TheScaleX mul /x exch def\n");
  fprintf(fp, "dup\n");
  fprintf(fp, "stringwidth\n");
  fprintf(fp, "pop\n");
  fprintf(fp, "/sh TheFontSize TheScaleFont mul def\n");
  fprintf(fp, "/sw exch def\n");
  fprintf(fp, "x sw sub\n");
  fprintf(fp, "y sh 2 div sub\n");
  fprintf(fp, "moveto\n");
  fprintf(fp, "show\n");
  fprintf(fp, "} def\n");
  fprintf(fp, "\n");
  fprintf(fp, "/leftText {\n");
  fprintf(fp, "%% called as: text x y leftText\n");
  fprintf(fp, "TheScaleY mul /y exch def\n");
  fprintf(fp, "TheScaleX mul /x exch def\n");
  fprintf(fp, "dup\n");
  fprintf(fp, "stringwidth\n");
  fprintf(fp, "pop\n");
  fprintf(fp, "/sh TheFontSize TheScaleFont mul def\n");
  fprintf(fp, "/sw exch def\n");
  fprintf(fp, "x\n");
  fprintf(fp, "y sh 2 div sub\n");
  fprintf(fp, "moveto\n");
  fprintf(fp, "show\n");
  fprintf(fp, "} def\n");
  fprintf(fp, "\n");
  fprintf(fp, "/setFontInfo {\n");
  fprintf(fp, "%% called as: font-name font-size setFontInfo\n");
  fprintf(fp, "/TheFontSize exch def\n");
  fprintf(fp, "/TheFontName exch def\n");
  fprintf(fp, "TheFontName findfont TheFontSize TheScaleFont mul scalefont setfont\n"
);
  fprintf(fp, "} def\n");
```

```
  fprintf(fp, "\n");
  fprintf(fp, "/setAxesScale {\n");
  fprintf(fp, "%% called as: x y scale-x scale-y setAxesScale\n");
  fprintf(fp, "/scaley exch def\n");
  fprintf(fp, "/scalex exch def\n");
  fprintf(fp, "TheScaleY mul /y exch def\n");
  fprintf(fp, "TheScaleX mul /x exch def\n");
  fprintf(fp, "gsave\n");
  fprintf(fp, "x y translate\n");
  fprintf(fp, "/TheScaleX TheScaleX scalex mul def\n");
  fprintf(fp, "/TheScaleY TheScaleY scaley mul def\n");
  fprintf(fp, "} def\n");
  fprintf(fp, "\n");
  fprintf(fp, "/restoreAxesScale {\n");
  fprintf(fp, "%% called as: x y scale-x scale-y restoreAxesScale\n");
  fprintf(fp, "/scaley exch def\n");
  fprintf(fp, "/scalex exch def\n");
  fprintf(fp, "TheScaleY mul /y exch def\n");
  fprintf(fp, "TheScaleX mul /x exch def\n");
  fprintf(fp, "grestore\n");
  fprintf(fp, "/TheScaleX TheScaleX scalex div def\n");
  fprintf(fp, "/TheScaleY TheScaleY scaley div def\n");
  fprintf(fp, "} def\n");
  fprintf(fp, "\n");
  fprintf(fp, "%% set initial/default values\n");
  fprintf(fp, "/Helvetica l2 setFontInfo\n");
  fprintf(fp, "0.5 setlinewidth\n");
  fprintf(fp, "\n");
  fprintf(fp, "%% set translation to whole paper\n");

/*#ifdef PS_DEBUG*/
#if 1
  fprintf(fp, "%.5g inch %.5g inch %.5g inch %.5g inch rectangle\n",
          0.0, 0.0, paperWidth, paperHeight);
#endif

  /* 用紙全体の座標系の設定 */
  if (data->orientation == PAPER_LANDSCAPE) {
    fprintf(fp, "%% paper orientation is landscape.\n");
    fprintf(fp, "%.5g inch %.5g inch %.5g inch %.5g inch setAxesScale\n",
            paperWidth - rightMargin,
            bottomMargin,
            paperHeight - topMargin - bottomMargin,
            paperWidth - rightMargin - leftMargin);
    fprintf(fp, "90 rotate\n");
  }
  else {
    fprintf(fp, "%% paper orientation is portrait.\n");
    fprintf(fp, "%.5g inch %.5g inch %.5g inch %.5g inch setAxesScale\n",
            leftMargin,
            bottomMargin,
            paperWidth - rightMargin - leftMargin,
            paperHeight - topMargin - bottomMargin);
  }

  fprintf(fp, "\n");

  for (i = 0;i < data->nGraphs;i++) {
    /* 個々のグラフの相対位置、サイズを求める */
    x = (float)data->winX[i]/data->mainWinWidth;
    y = (float)(data->mainWinHeight-data->winY[i])/data->mainWinHeight;
    width = (float)data->winWidth[i]/data->mainWinWidth;
    height = (float)data->winHeight[i]/data->mainWinHeight;

    /* 個々のグラフ座標系の設定 */
    fprintf(fp, "%%%% Graph %d\n", i+1);
    fprintf(fp, "%%%%\n");
    fprintf(fp, "%.5g %.5g %.5g %.5g setAxesScale\n",
            x, y, width, height);
    fprintf(fp, "\n");

    /* グラフの描画出力 */
    GrPrint(gGraphNo[i], fp);

    /* 用紙全体の座標系へ戻す */
    fprintf(fp, "\n");
    fprintf(fp, "%.5g %.5g %.5g %.5g restoreAxesScale\n",
            x, y, width, height);
    fprintf(fp, "\n");
  }

  /* タイトルウィンドウの描画 */
  fprintf(fp, "%%%%\n");
  fprintf(fp, "%%%% Title\n");
  fprintf(fp, "%%%%\n");
  x = (float)data->titleWinX/data->mainWinWidth;
  y = (float)(data->mainWinHeight-data->titleWinY)/data->mainWinHeight;
  width = (float)data->titleWinWidth/data->mainWinWidth;
  height = -(float)data->titleWinHeight/data->mainWinHeight;
  grPSDrawTitleCenter(fp, gGraphNo[0], data->title,
                      &x, &y, &width, &height);

  /* 中心点の出力 */
  fprintf(fp, "%%%%\n");
  fprintf(fp, "%%%% Center mark\n");
  fprintf(fp, "%%%%\n");
  x = (float)data->titleWinX/data->mainWinWidth;
  y = (float)(data->mainWinHeight-data->titleWinY)/data->mainWinHeight;
  width = (float)data->titleWinWidth/data->mainWinWidth;
  height = -(float)data->titleWinHeight/data->mainWinHeight;
  grPSSetFont(fp, "Helvetica", NULL, NULL, 10);
  if (data->orientation == PAPER_LANDSCAPE) {
    fprintf(fp, "(%s) %.5g %.5g %.5g %.5g centerText\n",
            "+", 0.0, 1.0, 1.0, 0.08);
  }
  else {
    fprintf(fp, "(%s) %.5g %.5g %.5g %.5g centerText\n",
            "+", 0.0, 0.0, -0.08, 1.0);
  }

  /* Prologの出力 */
  fprintf(fp, "\n");
  fprintf(fp, "showpage\n");
  fclose(fp);

  return(0);
}

/*
 * プロットウィンドウの大きさの再設定
 */
int SetPlotWindowSize(DATA *data)
{
  int i;
  ulong wPitch, hPitch;
  int wCount, hCount;
  int offset;
  XWindowChanges attr;
#ifdef DEBUG
  fprintf(stderr, "SetPlotWindowSize()\n");
#endif
```

```
offset = DEFAULT_TITLE_WINDOW_HEIGHT*data->mainWinHeight;

/* 横方向表示の場合 */
if (data->orientation == PAPER_LANDSCAPE) {
  switch (data->nGraphs) {
  case 1:
    wCount = 1; hCount = 1; break;
  case 2:
    wCount = 2; hCount = 1; break;
  case 3:
    wCount = 3; hCount = 1; break;
  case 4:
    wCount = 2; hCount = 2; break;
  case 5:
  case 6:
    wCount = 3; hCount = 2; break;
  case 7:
  case 8:
    wCount = 4; hCount = 2; break;
  case 9:
    wCount = 3; hCount = 3; break;
  default:
    wCount = 4; hCount = 3; break;
  }
}
/* 縦方向表示の場合 */
else {
  switch (data->nGraphs) {
  case 1:
    wCount = 1; hCount = 1; break;
  case 2:
    wCount = 1; hCount = 2; break;
  case 3:
    wCount = 1; hCount = 3; break;
  case 4:
    wCount = 1; hCount = 4; break;
  case 5:
  case 6:
    wCount = 2; hCount = 3; break;
  case 7:
  case 8:
    wCount = 2; hCount = 4; break;
  case 9:
    wCount = 3; hCount = 3; break;
  default:
    wCount = 3; hCount = 4; break;
  }
}

wPitch = (ulong)data->mainWinWidth/wCount;
hPitch = (ulong)(data->mainWinHeight-offset)/hCount;

for (i = 0;i < data->nGraphs;i++) {
  attr.x = (i % wCount) * wPitch;
  attr.y = offset + (int)(i / wCount) * hPitch;
  attr.width = wPitch;
  attr.height = hPitch;
  if (data->mode == X_DRAW_MODE)
    XConfigureWindow(data->display, data->winId[i],
                     CWX | CWY | CWWidth | CWHeight,
                     &attr);
  data->winX[i] = attr.x;
  data->winY[i] = attr.y;
  data->winWidth[i] = attr.width;
  data->winHeight[i] = attr.height;
}

/* タイトルウィンドウのリサイズ */
attr.x = 0;
attr.y = 0;
attr.width = data->mainWinWidth;
attr.height = offset;
if (data->mode == X_DRAW_MODE)
  XConfigureWindow(data->display, data->titleWinId,
                   CWX | CWY | CWWidth | CWHeight,
                   &attr);
data->titleWinX = attr.x;
data->titleWinY = attr.y;
data->titleWinWidth = attr.width;
data->titleWinHeight = attr.height;

return(0);
}

/*
 * ウィンドウの生成
 */
int CreatePlotWindows(DATA *data)
{
  ulong forePixel;
  ulong backPixel;
  ulong width, height;
  int screen;
  int i;
  int margin = 0;

#ifdef DEBUG
  fprintf(stderr, "CreatePlotWindows()\n");
  margin = 1;
#endif
  /* ウィンドウのデフォルト値の設定 */
  if (data->orientation == PAPER_LANDSCAPE) {
    width = DEFAULT_MAIN_WINDOW_WIDTH;
    height = DEFAULT_MAIN_WINDOW_HEIGHT;
  }
  else {
    width = DEFAULT_MAIN_WINDOW_HEIGHT;
    height = DEFAULT_MAIN_WINDOW_WIDTH;
  }
  data->mainWinWidth = width;
  data->mainWinHeight = height;

  /* PS出力モードの場合、リターン */
  if (data->mode == PS_ONLY_MODE)
    return(0);

  screen = DefaultScreen(data->display);
  forePixel = BlackPixel(data->display, screen);
  backPixel = WhitePixel(data->display, screen);

  /* メインウィンドウの生成 */
  data->mainWinId = XCreateSimpleWindow(data->display,
                                        DefaultRootWindow(data->display),
                                        0, 0, width, height,
                                        0, forePixel, backPixel);

  /* イベントの要請 */
  XSelectInput(data->display, data->mainWinId,
               ExposureMask | KeyPressMask |
               StructureNotifyMask | SubstructureNotifyMask);
```

```
  data->titleWinId = XCreateSimpleWindow(data->display,
                                         data->mainWinId,
                                         0, 0, width, height,
                                         margin, forePixel, backPixel);
  /* イベントの要請 */
  XSelectInput(data->display, data->titleWinId,
               ExposureMask | KeyPressMask);
  XMapRaised(data->display, data->mainWinId);

  for (i = 0;i < MAX_GRAPH_COUNT;i++) {
    /* ウィンドウの生成 */
    data->winId[i] = XCreateSimpleWindow(data->display,
                                         data->mainWinId,
                                         0, 0, 1, 1,
                                         margin, forePixel, backPixel);
    /* イベントの要請 */
    XSelectInput(data->display, data->winId[i],
                 ExposureMask | KeyPressMask);
    XClearWindow(data->display, data->winId[i]);

  }
  /* スタック順序の変更 */
  XRestackWindows(data->display, data->winId, MAX_GRAPH_COUNT);

  /* プロットウィンドウのマップ */
  XMapSubwindows(data->display, data->mainWinId);

  return(0);
}
```

```
/*
 * ACR PLOT TOOL Version 1.0
 *    (c) Copyright 1997
 *    ATR Adaptive Communications Research Laboratories
 *    All Rights Reserved
 */


/*
 * graph.c - グラフ描画関数
 */

#include <stdio.h>
#include <varargs.h>
#include <math.h>
#include <string.h>
#include <X11/X.h>
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <X11/Intrinsic.h>
#include "graph.h"

/*****************************************************************************
 *        Name: GrInit
 *    Function: グラフ情報の初期化
 *    Argument: なし
 *      Return: なし
 * Description:
 *      グラフハンドルの初期化を行なう。
 *
 *****************************************************************************/
int GrInit(void)
{
    int i;
#ifdef DEBUG
    fprintf(stderr, "GrInit()\n");fflush(stderr);
#endif
    for (i = 0;i < GR_MAX_GRAPH;i++)
        gGraph[i].on = 0;
    return(0);
}

/*****************************************************************************
 *        Name: GrOpen
 *    Function: グラフのオープン
 *    Argument:
 *              Display *disp   I       ディスプレイ構造体
 *              Window  win     I       ウィンドウID
 *
 *      Return:
 *              >=0  割り当てたグラフハンドル
 *              < 0  失敗
 * Description:
 *      1 空いているグラフハンドルを割り当てる。
 *      2 割り当てたグラフ情報の初期化を行なう。
 *
 *****************************************************************************/
Graph GrOpen(Display *disp, Window win)
{
    int         i;                      /* ループカウンタ */
    GrAttr      *p;                     /* カレントグラフ情報 */

#ifdef DEBUG
    fprintf(stderr, "GrOpen()\n");fflush(stderr);
#endif
    /* 空いているグラフ情報を捜す */
    for (i = 0;i < GR_MAX_GRAPH;i++) {
        if (gGraph[i].on == FALSE)
            break;
    }
    if (i >= GR_MAX_GRAPH)
        return(-1);

    p = gGraph+i;
    p->on               = TRUE;
    p->disp             = disp;
    p->win              = win;
    p->x                = GR_POS_DEFAULT_X;
    p->y                = GR_POS_DEFAULT_Y;
    p->width            = GR_POS_DEFAULT_WIDTH;
    p->height           = GR_POS_DEFAULT_HEIGHT;
    strcpy(p->line_style, GR_DEFAULT_LINE_STYLE);
    p->line_width       = GR_DEFAULT_LINE_WIDTH;
    p->dot_size         = 1.0;
    p->xdata            = NULL;
    p->xdatatype        = GR_FLOAT;
    p->xrange[0]        = 0;
    p->xrange[1]        = -1;
    p->ydata            = NULL;
    p->ydatatype        = GR_FLOAT;
    p->datacount        = 0;
    p->yrange[0]        = 0;
    p->yrange[1]        = -1;
    if (gPSOnlyMode == False) {
      p->foregc           = XCreateGC(p->disp, p->win, 0, 0);
      p->backgc           = XCreateGC(p->disp, p->win, 0, 0);
    }
    p->box              = FALSE;
    p->aspect_ratio     = -1;
    p->xaxis            = TRUE;
    p->yaxis            = TRUE;

    p->fontname         = strdup(GR_DEFAULT_FONTNAME);
    p->fontbold         = strdup(GR_DEFAULT_FONTBOLD);
    p->fontitalic       = strdup(GR_DEFAULT_FONTITALIC);
    p->fontsize         = GR_DEFAULT_FONTSIZE;
    if (gPSOnlyMode == False &&
        grLoadFontByType(p->disp, p->fontname,
                         p->fontbold, p->fontsize, p->fontitalic,
                         &p->font)) {
        fprintf(stderr, "Error: Can't load font\n");
        return(-1);
    }
    p->xticks           = NULL;
    p->num_xticks       = 0;
    p->yticks           = NULL;
    p->num_yticks       = 0;
    p->xlabel           = NULL;
    p->ylabel           = NULL;
    p->xtick_marks      = NULL;
    p->num_xtick_marks  = 0;
    p->ytick_marks      = NULL;
    p->num_ytick_marks  = 0;
    p->xticks_minor     = NULL;
    p->num_xticks_minor = 0;
    p->yticks_minor     = NULL;
    p->num_yticks_minor = 0;
    p->xtick_scale      = NULL;
    p->ytick_scale      = NULL;
    p->decoration_color = NULL;
    p->decoration_type  = 1;
    if (gPSOnlyMode == False)
```

```
  p->decogc           = XCreateGC(p->disp, p->win, 0, 0);

/* タイトルフォント */
p->title_fontname   = strdup(GR_DEFAULT_TITLE_FONTNAME);
p->title_fontbold   = strdup(GR_DEFAULT_TITLE_FONTBOLD);
p->title_fontitalic = strdup(GR_DEFAULT_TITLE_FONTITALIC);
p->title_fontsize   = GR_DEFAULT_TITLE_FONTSIZE;
if (gPSOnlyMode == False &&
    grLoadFontByType(p->disp,
                     p->title_fontname,
                     p->title_fontbold,
                     p->title_fontsize,
                     p->title_fontitalic,
                     &p->title_font)) {
    fprintf(stderr, "Error: Can't load font\n");
    return(-1);
}
if (gPSOnlyMode == False) {
  p->title_gc         = XCreateGC(p->disp, p->win, 0, 0);
  p->title_color      = NULL;
  p->title_pixel      = BlackPixel(p->disp, 0);
  XSetFont(p->disp, p->title_gc, p->title_font->fid);
  XSetForeground(p->disp, p->title_gc, p->title_pixel);
}
else
  p->title_color      = NULL;

/* ラベルフォント */
p->label_fontname   = strdup(GR_DEFAULT_LABEL_FONTNAME);
p->label_fontbold   = strdup(GR_DEFAULT_LABEL_FONTBOLD);
p->label_fontitalic = strdup(GR_DEFAULT_LABEL_FONTITALIC);
p->label_fontsize   = GR_DEFAULT_LABEL_FONTSIZE;
if (gPSOnlyMode == False &&
    grLoadFontByType(p->disp,
                     p->label_fontname,
                     p->label_fontbold,
                     p->label_fontsize,
                     p->label_fontitalic,
                     &p->label_font)) {
    fprintf(stderr, "Error: Can't load font\n");
    return(-1);
}
if (gPSOnlyMode == False) {
  p->label_gc         = XCreateGC(p->disp, p->win, 0, 0);
  p->label_color      = NULL;
  p->label_pixel      = BlackPixel(p->disp, 0);
  XSetFont(p->disp, p->label_gc, p->label_font->fid);
  XSetForeground(p->disp, p->label_gc, p->label_pixel);
}
else
  p->label_color      = NULL;

/* 目盛りフォント */
p->tick_fontname    = strdup(GR_DEFAULT_TICK_FONTNAME);
p->tick_fontbold    = strdup(GR_DEFAULT_TICK_FONTBOLD);
p->tick_fontitalic  = strdup(GR_DEFAULT_TICK_FONTITALIC);
p->tick_fontsize    = GR_DEFAULT_TICK_FONTSIZE;
if (gPSOnlyMode == False &&
    grLoadFontByType(p->disp,
                     p->tick_fontname,
                     p->tick_fontbold,
                     p->tick_fontsize,
                     p->tick_fontitalic,
                     &p->tick_font)) {
    fprintf(stderr, "Error: Can't load font\n");
    return(-1);
}
if (gPSOnlyMode == False) {
  p->tick_gc          = XCreateGC(p->disp, p->win, 0, 0);
  p->tick_color       = NULL;
  p->tick_pixel       = BlackPixel(p->disp, 0);
  XSetFont(p->disp, p->tick_gc, p->tick_font->fid);
  XSetForeground(p->disp, p->tick_gc, p->tick_pixel);
}
else
  p->tick_color       = NULL;

/* 目盛り     /* 目盛りフォント */
p->tick_fontname    = strdup(GR_DEFAULT_TICK_FONTNAME);
p->tick_fontbold    = strdup(GR_DEFAULT_TICK_FONTBOLD);
p->tick_fontitalic  = strdup(GR_DEFAULT_TICK_FONTITALIC);
p->tick_fontsize    = GR_DEFAULT_TICK_FONTSIZE;
if (gPSOnlyMode == False &&
    grLoadFontByType(p->disp,
                     p->tick_fontname,
                     p->tick_fontbold,
                     p->tick_fontsize,
                     p->tick_fontitalic,
                     &p->tick_font)) {
    fprintf(stderr, "Error: Can't load font\n");
    return(-1);
}
if (gPSOnlyMode == False) {
  p->tick_gc          = XCreateGC(p->disp, p->win, 0, 0);
  p->tick_color       = NULL;
  p->tick_pixel       = BlackPixel(p->disp, 0);
  XSetFont(p->disp, p->tick_gc, p->tick_font->fid);
  XSetForeground(p->disp, p->tick_gc, p->tick_pixel);
}
else
  p->tick_color       = NULL;

/* 目盛り(上付)フォント */
p->ticksub_fontname = strdup(GR_DEFAULT_TICKSUB_FONTNAME);
p->ticksub_fontbold = strdup(GR_DEFAULT_TICKSUB_FONTBOLD);
p->ticksub_fontitalic = strdup(GR_DEFAULT_TICKSUB_FONTITALIC);
p->ticksub_fontsize = GR_DEFAULT_TICKSUB_FONTSIZE;
if (gPSOnlyMode == False &&
    grLoadFontByType(p->disp,
                     p->ticksub_fontname,
                     p->ticksub_fontbold,
                     p->ticksub_fontsize,
                     p->ticksub_fontitalic,
                     &p->ticksub_font)) {
    fprintf(stderr, "Error: Can't load font\n");
    return(-1);
}
if (gPSOnlyMode == False) {
  p->ticksub_gc       = XCreateGC(p->disp, p->win, 0, 0);
  p->ticksub_color    = NULL;
  p->ticksub_pixel    = BlackPixel(p->disp, 0);
  XSetFont(p->disp, p->ticksub_gc, p->ticksub_font->fid);
  XSetForeground(p->disp, p->ticksub_gc, p->ticksub_pixel);
}
else
  p->ticksub_color    = NULL;

p->draw_mode        = GR_MODE_NORMAL;
p->draw_size        = 0;
p->draw_ptr         = 0;
p->draw_buffer      = NULL;
p->hold             = FALSE;
```

```
    p->max_exp          = GR_DEFAULT_MAX_EXP;
    p->graph_type       = GR_DEFAULT_GRAPH_TYPE;

    /* GC */
    if (gPSOnlyMode == False) {
      XSetForeground(p->disp, p->foregc, BlackPixel(p->disp, 0));
      XSetForeground(p->disp, p->backgc, WhitePixel(p->disp, 0));
      XSetFont(p->disp, p->foregc, p->font->fid);
    }

    return(i);
}

/*****************************************************************************
 *        Name: GrSet
 *    Function: グラフ情報の設定
 *    Argument:
 *              Graph gr        I       グラフハンドル
 *              (ResourceID)    I       リソースID
 *              (ResourceValue) I       リソース値
 *      Return:
 *              0  成功
 *             -1  失敗
 * Description:
 *
 *****************************************************************************/
int GrSet(gr, va_alist)
Graph gr;
va_dcl
{
    GrAttr      *p;
    va_list     ap;
    int         prop;
    float       *xdata, *ydata;
    float       *xrange, *yrange;
    int         isfont = FALSE;
    int         istitle_font = FALSE;
    int         islabel_font = FALSE;
    int         istick_font = FALSE;
    int         isticksub_font = FALSE;
    int         is_draw_mode = FALSE;
    int         is_xrange = FALSE;
    int         is_yrange = FALSE;
    char        *str;

#ifdef DEBUG
    fprintf(stderr, "GrSet(%d)\n", gr);fflush(stderr);
#endif
    if (gr < 0 || gr >= GR_MAX_GRAPH || !gGraph[gr].on) {
        return(-1);
    }

    p = gGraph+gr;
    va_start(ap);
    xdata = ydata = NULL;
    while (1) {
        switch (prop = va_arg(ap, int)) {
        case NULL:
            goto next;
            break;
        case GR_X:
            p->x = va_arg(ap, float);
            break;
        case GR_Y:
            p->y = va_arg(ap, float);
            break;
```

```
        case GR_WIDTH:
            p->width = va_arg(ap, float);
            break;
        case GR_HEIGHT:
            p->height = va_arg(ap, float);
            break;
        case GR_LINE_STYLE:
            strcpy(p->line_style, va_arg(ap, char *));
            break;
        case GR_LINE_WIDTH:
            p->line_width = va_arg(ap, float);
            break;
        case GR_DOT_SIZE:
            { float *f;
              f = va_arg(ap, float *);
              p->dot_size = *f;
            }
            break;
        case GR_XDATA:
            xdata = va_arg(ap, float *);
            break;
        case GR_XDATATYPE:
            p->xdatatype = va_arg(ap, int);
            break;
        case GR_YDATA:
            ydata = va_arg(ap, float *);
            break;
        case GR_YDATATYPE:
            p->ydatatype = va_arg(ap, int);
            break;
        case GR_DATACOUNT:
            p->datacount = va_arg(ap, int);
            break;
        case GR_TITLE:
            p->title = strdup(va_arg(ap, char *));
            break;
        case GR_WINDOW:
            p->win = va_arg(ap, Window);
            break;
        case GR_BOX:
            p->box = va_arg(ap, int);
            break;
        case GR_ASPECT_RATIO:
            { float *f;
              f = va_arg(ap, float *);
              p->aspect_ratio = *f;
            }
            break;
        case GR_XTICKS:
            p->xticks = va_arg(ap, float *);
            break;
        case GR_NUM_XTICKS:
            p->num_xticks = va_arg(ap, int);
            break;
        case GR_YTICKS:
            p->yticks = va_arg(ap, float *);
            break;
        case GR_NUM_YTICKS:
            p->num_yticks = va_arg(ap, int);
            break;
        case GR_XLABEL:
            p->xlabel = strdup(va_arg(ap, char *));
            break;
        case GR_YLABEL:
            p->ylabel = strdup(va_arg(ap, char *));
            break;
```

```
case GR_DECORATION_COLOR:
    str = va_arg(ap, char *);
    if (str && *str != '\0') {
      p->decoration_color = strdup(str);
      if (gPSOnlyMode == False)
        grAllocHighLowColor(p->disp,
                            p->decoration_color,
                            p->decoration_pixel);
    }
    break;
case GR_XRANGE:
    xrange = va_arg(ap, float *);
    is_xrange = TRUE;
    break;
case GR_YRANGE:
    yrange = va_arg(ap, float *);
    is_yrange = TRUE;
    break;
case GR_FONTNAME:
    p->fontname = strdup(va_arg(ap, char *));
    isfont = TRUE;
    break;
case GR_FONTBOLD:
    p->fontbold = strdup(va_arg(ap, char *));
    isfont = TRUE;
    break;
case GR_FONTITALIC:
    p->fontitalic = strdup(va_arg(ap, char *));
    isfont = TRUE;
    break;
case GR_FONTSIZE:
    p->fontsize = va_arg(ap, int);
    isfont = TRUE;
    break;
case GR_TITLE_FONTNAME:
    p->title_fontname = strdup(va_arg(ap, char *));
    istitle_font = TRUE;
    break;
case GR_TITLE_FONTBOLD:
    p->title_fontbold = strdup(va_arg(ap, char *));
    istitle_font = TRUE;
    break;
case GR_TITLE_FONTITALIC:
    p->title_fontitalic = strdup(va_arg(ap, char *));
    istitle_font = TRUE;
    break;
case GR_TITLE_FONTSIZE:
    p->title_fontsize = va_arg(ap, int);
    istitle_font = TRUE;
    break;
case GR_TITLE_COLOR:
    {
        Colormap        cmap;
        char            *name;
        XColor          ci, cr;

        if (gPSOnlyMode == False)
          cmap = DefaultColormap(p->disp, 0);
        name = strdup(va_arg(ap, char *));
        if (p->title_color) {
            Free(p->title_color);
            if (gPSOnlyMode == False)
              XFreeColors(p->disp, cmap, &p->title_pixel, 1, 0);
        }
        if (!*name) {
            Free(name);
            p->title_color = NULL;
            if (gPSOnlyMode == False)
              p->title_pixel = BlackPixel(p->disp, 0);
            break;
        }
        p->title_color = name;
        if (gPSOnlyMode == False) {
          XAllocNamedColor(p->disp, cmap, p->title_color, &ci, &cr);
          p->title_pixel = cr.pixel;
          XSetForeground(p->disp, p->title_gc, p->title_pixel);
        }
    }
    break;
case GR_LABEL_FONTNAME:
    p->label_fontname = strdup(va_arg(ap, char *));
    islabel_font = TRUE;
    break;
case GR_LABEL_FONTBOLD:
    p->label_fontbold = strdup(va_arg(ap, char *));
    islabel_font = TRUE;
    break;
case GR_LABEL_FONTITALIC:
    p->label_fontitalic = strdup(va_arg(ap, char *));
    islabel_font = TRUE;
    break;
case GR_LABEL_FONTSIZE:
    p->label_fontsize = va_arg(ap, int);
    islabel_font = TRUE;
    break;
case GR_LABEL_COLOR:
    {
        Colormap        cmap;
        char            *name;
        XColor          ci, cr;

        if (gPSOnlyMode == False)
          cmap = DefaultColormap(p->disp, 0);
        name = strdup(va_arg(ap, char *));
        if (p->label_color) {
            Free(p->label_color);
            if (gPSOnlyMode == False)
              XFreeColors(p->disp, cmap, &p->label_pixel, 1, 0);
        }
        if (!*name) {
            Free(name);
            p->label_color = NULL;
            if (gPSOnlyMode == False)
              p->label_pixel = BlackPixel(p->disp, 0);
            break;
        }
        p->label_color = name;
        if (gPSOnlyMode == False) {
          XAllocNamedColor(p->disp, cmap, p->label_color, &ci, &cr);
          p->label_pixel = cr.pixel;
          XSetForeground(p->disp, p->label_gc, p->label_pixel);
        }
    }
    break;
case GR_TICK_FONTNAME:
    p->tick_fontname = strdup(va_arg(ap, char *));
    istick_font = TRUE;
    break;
case GR_TICK_FONTBOLD:
    p->tick_fontbold = strdup(va_arg(ap, char *));
    istick_font = TRUE;
    break;
```

```
case GR_TICK_FONTITALIC:
    p->tick_fontitalic = strdup(va_arg(ap, char *));
    istick_font = TRUE;
    break;
case GR_TICK_FONTSIZE:
    p->tick_fontsize = va_arg(ap, int);
    istick_font = TRUE;
    break;
case GR_TICK_COLOR:
    {
        Colormap        cmap;
        char            *name;
        XColor          ci, cr;

        if (gPSOnlyMode == False)
          cmap = DefaultColormap(p->disp, 0);
        name = strdup(va_arg(ap, char *));
        if (p->tick_color) {
            Free(p->tick_color);
            if (gPSOnlyMode == False)
              XFreeColors(p->disp, cmap, &p->tick_pixel, 1, 0);
        }
        if (!*name) {
            Free(name);
            p->tick_color = NULL;
            if (gPSOnlyMode == False)
              p->tick_pixel = BlackPixel(p->disp, 0);
            break;
        }
        p->tick_color = name;
        if (gPSOnlyMode == False) {
          XAllocNamedColor(p->disp, cmap, p->tick_color, &ci, &cr);
          p->tick_pixel = cr.pixel;
          XSetForeground(p->disp, p->tick_gc, p->tick_pixel);
        }
    }
    break;
case GR_TICKSUB_FONTNAME:
    p->ticksub_fontname = strdup(va_arg(ap, char *));
    isticksub_font = TRUE;
    break;
case GR_TICKSUB_FONTBOLD:
    p->ticksub_fontbold = strdup(va_arg(ap, char *));
    isticksub_font = TRUE;
    break;
case GR_TICKSUB_FONTITALIC:
    p->ticksub_fontitalic = strdup(va_arg(ap, char *));
    isticksub_font = TRUE;
    break;
case GR_TICKSUB_FONTSIZE:
    p->ticksub_fontsize = va_arg(ap, int);
    isticksub_font = TRUE;
    break;
case GR_TICKSUB_COLOR:
    {
        Colormap        cmap;
        char            *name;
        XColor          ci, cr;

        if (gPSOnlyMode == False)
          cmap = DefaultColormap(p->disp, 0);
        name = strdup(va_arg(ap, char *));
        if (p->ticksub_color) {
            Free(p->ticksub_color);
            if (gPSOnlyMode == False)
              XFreeColors(p->disp, cmap, &p->ticksub_pixel, 1, 0);
            }
            if (!*name) {
                Free(name);
                p->ticksub_color = NULL;
                if (gPSOnlyMode == False)
                  p->ticksub_pixel = BlackPixel(p->disp, 0);
                break;
            }
            p->ticksub_color = name;
            if (gPSOnlyMode == False) {
              XAllocNamedColor(p->disp, cmap, p->ticksub_color, &ci, &cr);
              p->ticksub_pixel = cr.pixel;
              XSetForeground(p->disp, p->ticksub_gc, p->ticksub_pixel);
            }
        }
        break;
    case GR_DRAW_MODE:
        p->draw_mode = va_arg(ap, int);
        is_draw_mode = TRUE;
        break;
    case GR_DRAW_SIZE:
        {
            int     size;
            size = va_arg(ap, int);
            if (size < 0)
                break;
            if (p->draw_buffer)
                Free(p->draw_buffer);
            p->draw_buffer = (Number *)malloc(sizeof(Number)*size);
            p->draw_size = size;
            p->draw_ptr = 0;
            break;
        }
    case GR_HOLD:
        p->hold = va_arg(ap, int);
        break;
    case GR_MAX_EXP:
        p->max_exp = va_arg(ap, int);
        break;
    case GR_GRAPH_TYPE:
        p->graph_type = va_arg(ap, int);
        break;
    default:
        break;
    }
}

next:
    va_end(ap);

    /* データ変更 */
    if (xdata) {
        int     len = sizeof(Number)*p->datacount;
        Free(p->xdata);
        p->xdata = (Number *)malloc(len);
        memcpy(p->xdata, xdata, len);
    }
    if (ydata) {
        int     len;
        int     rem;
        if (p->ydata)
            Free(p->ydata);
        switch (p->draw_mode) {
        case GR_MODE_NORMAL:
            len = sizeof(Number)*p->datacount;
            p->ydata = (Number *)malloc(len);
```

```
            memcpy(p->ydata, ydata, len);
            break;
        case GR_MODE_FLOW:
            rem = p->datacount - p->draw_size;
            if (rem > 0) {
                p->datacount -= rem;
                ydata += rem;
            }
            rem = p->datacount - (p->draw_size - p->draw_ptr);
            if (rem < 0) {
                len = sizeof(Number)*(p->draw_ptr+1-rem);
                memcpy(p->draw_buffer, p->draw_buffer+rem, len);
                p->draw_ptr += rem;
            }
            len = sizeof(Number)*p->datacount;
            memcpy(p->draw_buffer+p->draw_ptr, ydata, len);
            p->draw_ptr += p->datacount;
            break;
        }
        if (p->hold == FALSE) {
            grFreeAxisAttributes(gr);
        }
    }

    /* フォント変更 */
    if (gPSOnlyMode == False && isfont == TRUE) {
      if (p->font)
        XFreeFont(p->disp, p->font);
      if (grLoadFontByType(p->disp, p->fontname,
                           p->fontbold, p->fontsize, p->fontitalic,
                           &p->font)) {
        fprintf(stderr, "Error: Can't load font\n");
        return(-1);
      }
      XSetFont(p->disp, p->foregc, p->font->fid);
    }
    /* タイトルフォント変更 */
    if (gPSOnlyMode == False && istitle_font == TRUE) {
      XFreeFont(p->disp, p->title_font);
      if (grLoadFontByType(p->disp, p->title_fontname,
                           p->title_fontbold,
                           p->title_fontsize,
                           p->title_fontitalic,
                           &p->title_font)) {
        fprintf(stderr, "Error: Can't load font\n");
        return(-1);
      }
      XSetFont(p->disp, p->title_gc, p->title_font->fid);
    }
    /* ラベルフォント変更 */
    if (gPSOnlyMode == False && islabel_font == TRUE) {
      if (p->label_font)
        XFreeFont(p->disp, p->label_font);
      if (grLoadFontByType(p->disp, p->label_fontname,
                           p->label_fontbold,
                           p->label_fontsize,
                           p->label_fontitalic,
                           &p->label_font)) {
        fprintf(stderr, "Error: Can't load font\n");
        return(-1);
      }
      XSetFont(p->disp, p->label_gc, p->label_font->fid);
    }
    /* 目盛りフォント変更 */
    if (gPSOnlyMode == False && istick_font == TRUE) {
      if (p->tick_font)
        XFreeFont(p->disp, p->tick_font);
      if (grLoadFontByType(p->disp, p->tick_fontname,
                           p->tick_fontbold,
                           p->tick_fontsize,
                           p->tick_fontitalic,
                           &p->tick_font)) {
        fprintf(stderr, "Error: Can't load font\n");
        return(-1);
      }
      XSetFont(p->disp, p->tick_gc, p->tick_font->fid);
    }
    /* 目盛り(上付)フォント変更 */
    if (gPSOnlyMode == False && isticksub_font == TRUE) {
      if (p->ticksub_font)
        XFreeFont(p->disp, p->ticksub_font);
      if (grLoadFontByType(p->disp, p->ticksub_fontname,
                           p->ticksub_fontbold,
                           p->ticksub_fontsize,
                           p->ticksub_fontitalic,
                           &p->ticksub_font)) {
        fprintf(stderr, "Error: Can't load font\n");
        return(-1);
      }
      XSetFont(p->disp, p->ticksub_gc, p->ticksub_font->fid);
    }

    /* 描画モード変更 */
    if (is_draw_mode == TRUE) {
    }

    if (is_xrange == TRUE) {
        p->xrange[0] = xrange[0];
        p->xrange[1] = xrange[1];
    }
    if (is_yrange == TRUE) {
        p->yrange[0] = yrange[0];
        p->yrange[1] = yrange[1];
    }

    return(0);
}

/*****************************************************************************
 *        Name: GrHold
 *    Function: グラフ情報の保持フラグのセット
 *    Argument:
 *              Graph gr        I       グラフハンドル
 *      Return:
 *              0  成功
 *             -1  失敗
 * Description:
 *
 *****************************************************************************/
int GrHold(Graph gr, int on)
{
    GrAttr      *p;
#ifdef DEBUG
    fprintf(stderr, "GrHold(%d)\n", gr);fflush(stderr);
#endif
    if (gr < 0 || gr >= GR_MAX_GRAPH || !gGraph[gr].on)
        return(-1);

    p = gGraph + gr;
    p->hold = on;
    return(0);
}
```

```
/*      8       16      24      32      40      48      56                  */
/*------+-------+-------+-------+-------+-------+-------+-------------------*/
/*****************************************************************************
 *        Name: GrPlot
 *    Function: グラフのプロット
 *    Argument:
 *              Graph gr        I       グラフハンドル
 *      Return:
 *               0  成功
 *              -1  失敗
 * Description:
 *              設定されたリソース値にしたがって描画を行なう。
 *              実際にウィンドウに描画されるのは、次の GrDraw 関数で行なう。
 *
 *****************************************************************************/
int GrPlot(Graph gr, Number *x, int nx, Number *y, int ny)
{
    GrAttr      *p;
    int         i;
#ifdef DEBUG
    fprintf(stderr, "GrPlot(%d)\n", gr);fflush(stderr);
#endif
    if (gr < 0 || gr >= GR_MAX_GRAPH || !gGraph[gr].on)
        return(-1);

    p = gGraph + gr;

    Free(p->xdata);
    Free(p->ydata);
    /* yデータ */
    p->datacount = ny;
    if (y && ny > 0) {
        int     len;
        int     rem;

        switch (p->draw_mode) {
        case GR_MODE_NORMAL:
            len = sizeof(Number)*ny;
            p->ydata = (Number *)malloc(len);
            memcpy(p->ydata, y, len);
            break;

        case GR_MODE_FLOW:
            rem = p->datacount - p->draw_size;
            if (rem > 0) {
                p->datacount -= rem;
                y += rem;
            }
            rem = p->datacount - (p->draw_size - p->draw_ptr);
            if (rem < 0) {
                len = sizeof(Number)*(p->draw_ptr+1-rem);
                memcpy(p->draw_buffer, p->draw_buffer+rem, len);
                p->draw_ptr += rem;
            }
            len = sizeof(Number)*p->datacount;
            memcpy(p->draw_buffer+p->draw_ptr, y, len);
            p->draw_ptr += p->datacount;
            break;
        }
    }

    /* xデータ */
    if (x && nx > 0) {
        int     len = sizeof(Number)*nx;
        p->xdata = (Number *)malloc(len);
        memcpy(p->xdata, x, len);
    }
    else {
#ifdef GRAPH_DEBUG
        fprintf(stderr, "  calc xdata\n");fflush(stderr);
#endif
        p->xdata = (Number *)malloc(sizeof(Number)*p->datacount);
        for (i = 0;i < p->datacount;i++)
            p->xdata[i] = i+1;
    }

    /* 保持フラグが立っていない場合、値の範囲をクリア */
    if (p->hold == FALSE) {
        grFreeAxisAttributes(gr);
        p->xrange[0] = p->yrange[0] = 0.0;
        p->xrange[1] = p->yrange[1] = -1.0;
    }

    return(0);
}

/*      8       16      24      32      40      48      56                  */
/*------+-------+-------+-------+-------+-------+-------+-------------------*/
/*****************************************************************************
 *        Name: GrDraw
 *    Function: グラフの描画
 *    Argument:
 *              Graph gr        I       グラフハンドル
 *      Return:
 *               0  成功
 *              -1  失敗
 * Description:
 *              ウィンドウへグラフの描画を行なう。
 *
 *****************************************************************************/
int GrDraw(Graph gr)
{
    GrAttr      *p;
    int         i;
    XPoint      *ps;

#ifdef GRAPH_DEBUG
    fprintf(stderr, "GrDraw(%d)\n", gr);fflush(stderr);
#endif
    if (gr < 0 || gr >= GR_MAX_GRAPH || !gGraph[gr].on)
        return(-1);

    p = gGraph + gr;
#ifdef GRAPH_DEBUG
    fprintf(stderr, "  hold: %d\n", p->hold);fflush(stderr);
#endif

    /* チェック */
    if (p->width <= 0 ||
        p->height <= 0 ||
        p->datacount < 0)
        return(-2);

    if (p->graph_type == GR_TYPE_METER) {
        return(grDrawMeter(gr));
    }

    /* 座標属性の計算 */
    if (p->hold == FALSE) {
        grCalcAxisAttributes(gr);
    }
```

```
    /* 描画領域の計算 */
    grCalcAxisOrigin(gr);
    grClearGraph(gr);

    if (p->hold == FALSE) {

        /* 装飾の描画 */
        if (p->decoration_color) {
            grDrawDecoration(gr);
        }
        else{
            GrClear(gr);
        }

        /* ボックス/座標軸の描画 */
#ifdef GRAPH_DEBUG
        fprintf(stderr, "  draw box/axis\n");fflush(stderr);
#endif
        if (p->box) {
            XDrawRectangle(p->disp, p->win, p->foregc,
                           p->Ox, p->Oy, p->Ow, p->Oh);
        }
        else {
            /* x座標軸 */
            XDrawLine(p->disp, p->win, p->foregc,
                      p->Ox-1, p->Oy-1 + p->Oh+2,
                      p->Ox-1 + p->Ow+2, p->Oy-1 + p->Oh+2);
            /* y座標軸 */
            XDrawLine(p->disp, p->win, p->foregc,
                      p->Ox-1, p->Oy-1 + p->Oh+2,
                      p->Ox-1, p->Oy-1);
            XDrawLine(p->disp, p->win, p->foregc,
                      p->Ox-1, p->Oy-1,
                      (int)(p->Ox-1 + p->Aw*GR_TICK_LENGTH/3),
                      p->Oy-1);
            XDrawLine(p->disp, p->win, p->foregc,
                      p->Ox-1 + p->Ow+2, p->Oy-1 + p->Oh+2,
                      p->Ox-1 + p->Ow+2,
                      (int)((p->Oy-1 + p->Oh+2)-p->Ah*GR_TICK_LENGTH/3));
        }

        /* x目盛り文字列の描画 */
        if (p->num_xtick_marks > 0) {
            int         i, x1, y1, x, y;
            float       rx, dx;

#ifdef GRAPH_DEBUG
            fprintf(stderr, "  draw xtick marks\n");fflush(stderr);
#endif
            rx = p->xrange[1] - p->xrange[0];
            if (rx == 0) dx = 0;
            else dx = p->Ow/rx;

            for (i = 0;i < p->num_xtick_marks;i++) {
                x1 = p->Ox + (p->xticks[i]-p->xrange[0])*dx;
                y1 = p->Oy + p->Oh + p->Ah*GR_POS_XTICKS_MARGIN;
                grGetCenterPos(gr, p->xtick_marks[i],
                               x1, y1, 0, GR_POS_XTICKS_HEIGHT, &x, &y);
                XDrawString(p->disp, p->win, p->tick_gc,
                            x, y, p->xtick_marks[i],
                            strlen(p->xtick_marks[i]));
            }
        }

        /* xスケールの描画 */
```

```
        if (p->xtick_scale) {
          int  x, y, xx, yy;
          char str[20], *ptr;
          unsigned int width, height;

          x = p->Rx + p->Rw;
          y = p->Oy + p->Oh + p->Ah*GR_POS_XTICKS_MARGIN
            + GR_POS_XTICKS_HEIGHT;

          strcpy(str, "x");
          strcat(str, p->xtick_scale);
          if ((ptr = strchr(str, '^')) != NULL)
            *ptr++ = '\0';
          grGetCenterPos(gr, str, x, y, -1, 0, &x, &y);
          XDrawString(p->disp, p->win, p->tick_gc,
                      x, y, str, strlen(str));
          if (ptr) {
            grTextWidth(p->disp, p->win, p->tick_gc,
                        str, strlen(str), &width, &height);
            XDrawString(p->disp, p->win, p->ticksub_gc,
                        x + width, y - height/2, ptr, strlen(ptr));
          }
        }

        /* y目盛り文字列の描画 */
        if (p->num_ytick_marks > 0) {
            int         i, x1, y1, x, y;
            float               ry, dy;

            ry = p->yrange[1] - p->yrange[0];
            if (ry == 0) dy = 0;
            else dy = p->Oh/ry;
            for (i = 0;i < p->num_ytick_marks;i++) {
                x1 = p->Ox - GR_POS_YTICKS_MARGIN;
                y1 = p->Oy + p->Oh - (p->yticks[i]-p->yrange[0])*dy;
                grGetCenterPos(gr, p->ytick_marks[i], x1, y1, -1, 0, &x, &y);
                XDrawString(p->disp, p->win, p->tick_gc,
                            x, y, p->ytick_marks[i],
                            strlen(p->ytick_marks[i]));
            }
        }

        /* タイトル文字列の描画 */
        if (p->title && *p->title) {
            int x, y;
            unsigned int        w, h;

            w = p->Aw;
            h = p->Ry;
            grGetCenterPos(gr, p->title, p->Ax, p->Ay, w, h, &x, &y);
            XDrawString(p->disp, p->win, p->title_gc,
                        x, y, p->title, strlen(p->title));
        }

        /* xラベル文字列の描画 */
        if (p->xlabel && *p->xlabel) {
            int x, y;
            unsigned int        w, h;

            w = p->Ow;
            h = (p->Ay + p->Ah) - (p->Ry + p->Rh);
            grGetCenterPos(gr, p->xlabel, p->Ox, p->Ry + p->Rh, w, h, &x, &y);
            XDrawString(p->disp, p->win, p->label_gc,
                        x, y, p->xlabel, strlen(p->xlabel));
        }
```

```
        /* yラベル文字列の描画 */
        if (p->ylabel && *p->ylabel) {
            int x, y;
            unsigned int        w, h;

            w = p->Rx - p->Ax;
            h = p->Oh;
/*          grGetCenterPos(gr, p->ylabel, p->Ax, p->Ay, h, w, &y, &x);
            grDrawStringDownToUp(p->disp, p->win, p->label_gc,
                                 x, p->Oy + p->Oh - (y - p->Ay),
                                 p->ylabel, strlen(p->ylabel));*/
            grGetCenterPos(gr, p->ylabel, 0, 0, h, w, &y, &x);
            grDrawStringDownToUp(p->disp, p->win, p->label_gc,
                                 p->Ax + w - x, p->Oy + p->Oh - y,
                                 p->ylabel, strlen(p->ylabel));
        }

        /* yスケールの描画 */
        if (p->ytick_scale) {
          int  x, y;
          char str[20], *ptr;
          unsigned int width, height;

          x = p->Ox;
          y = p->Ry;
          grGetCenterPos(gr, str, x, y, -1, 0, &x, &y);

          strcpy(str, "x");
          strcat(str, p->ytick_scale);
          if ((ptr = strchr(str, '^')) != NULL)
            *ptr++ = '\0';
          XDrawString(p->disp, p->win, p->tick_gc,
                      x, y, str, strlen(str));
          if (ptr) {
            grTextWidth(p->disp, p->win, p->tick_gc,
                        str, strlen(str), &width, &height);
            XDrawString(p->disp, p->win, p->ticksub_gc,
                        x + width, y - height/2, ptr, strlen(ptr));
          }
        }

    }

    /* x目盛りの描画 */
    if (p->xticks) {
        XSegment        seg[GR_TICK_MAX_ALL_TICKS];
        int             i, len;
        float   rx, dx;

#ifdef GRAPH_DEBUG
        fprintf(stderr, "  draw xticks\n");fflush(stderr);
#endif
        rx = p->xrange[1] - p->xrange[0];
        if (rx == 0) dx = 0;
        else dx = p->Ow/rx;

        /* Major目盛りの描画 */
        len = p->Ah*GR_TICK_LENGTH;
        if (len < 2) len = 2;
        for (i = 0;i < p->num_xticks;i++) {
            seg[i].x1 = seg[i].x2 = p->Ox + (p->xticks[i]-p->xrange[0])*dx;
            seg[i].y1 = p->Oy + p->Oh;
            seg[i].y2 = p->Oy + p->Oh - len;
        }
        XDrawSegments(p->disp, p->win, p->foregc, seg, p->num_xticks);
```

```
        /* Minor目盛りの描画 */
        len = p->Ah*GR_TICK_MINOR_LENGTH;
        if (len < 1) len = 1;
        if (p->num_xticks_minor > 0) {
            for (i = 0;i < p->num_xticks_minor;i++) {
                seg[i].x1 = seg[i].x2 = p->Ox +
                    (p->xticks_minor[i]-p->xrange[0])*dx;
                seg[i].y1 = p->Oy + p->Oh;
                seg[i].y2 = p->Oy + p->Oh - len;
            }
            XDrawSegments(p->disp, p->win, p->foregc,
                          seg, p->num_xticks_minor);
        }

    }

    /* y目盛りの描画 */
    if (p->yticks) {
        XSegment        seg[GR_TICK_MAX_ALL_TICKS];
        int             i, len;
        float           ry, dy;
#ifdef GRAPH_DEBUG
        fprintf(stderr, "  draw yticks\n");fflush(stderr);
#endif

        ry = p->yrange[1] - p->yrange[0];
        if (ry == 0) dy = 0;
        else dy = p->Oh/ry;

        /* Major目盛りの描画 */
        len = p->Aw*GR_TICK_LENGTH;
        if (len < 2) len = 2;
        for (i = 0;i < p->num_yticks;i++) {
            seg[i].y1 = seg[i].y2 = p->Oy + p->Oh
              - (p->yticks[i]-p->yrange[0])*dy;
            seg[i].x1 = p->Ox;
            seg[i].x2 = p->Ox + len;
        }
        XDrawSegments(p->disp, p->win, p->foregc, seg, p->num_yticks);

        /* Minor目盛りの描画 */
        len = p->Aw*GR_TICK_MINOR_LENGTH;
        if (len < 1) len = 1;
        if (p->num_yticks_minor > 0) {
          for (i = 0;i < p->num_yticks_minor;i++) {
            seg[i].y1 = seg[i].y2 = p->Oy + p->Oh
              - (p->yticks_minor[i]-p->yrange[0])*dy;
            seg[i].x1 = p->Ox;
            seg[i].x2 = p->Ox + len;
          }
          XDrawSegments(p->disp, p->win, p->foregc, seg,
                        p->num_yticks_minor);
        }
    }

    /* データの描画 */
    if (p->ydata) {
        XRectangle      rect[1];
        Number          x, y;

        /* データの相対座標から絶対座標への変換 */
        ps = (XPoint *)malloc(sizeof(XPoint)*(p->datacount));
        for (i = 0;i < p->datacount;i++) {
          x = (p->Ox + (p->xdata[i]-p->xrange[0])
                       /(p->xrange[1]-p->xrange[0])*p->Ow);
          y = (p->Oy + p->Oh - (p->ydata[i]-p->yrange[0])
```

```
                        /(p->yrange[1]-p->yrange[0])*p->0h);
          ps[i].x = rint(x);
          ps[i].y = rint(y);
        }
#ifdef GRAPH_DEBUG_EX
        {
            int         i;
            fprintf(stderr, "  xrange = [%g %g], yrange = [%g %g]\n",
                    p->xrange[0], p->xrange[1],
                    p->yrange[0], p->yrange[1]);
            fprintf(stderr, "  data = [");
            for (i = 0;i < p->datacount;i++)
                fprintf(stderr, "(%g,%g;%d,%d)",
                        p->xdata[i], p->ydata[i],
                        ps[i].x, ps[i].y);
            fprintf(stderr, "]\n");
        }
#endif
        /* プロット領域をクリップする */
        rect[0].x = p->0x-1;
        rect[0].y = p->0y-1;
        rect[0].width = p->0w+2;
        rect[0].height = p->0h+2;
        XSetClipRectangles(p->disp, p->foregc,
                           0, 0, rect, 1, Unsorted);

        /*
         * 線種による描きわけ
         */
        /* 直線 */
        if (!strcmp(p->line_style, GR_LINE_STYLE_LINE)) {
          XSetLineAttributes(p->disp, p->foregc, p->line_width,
                             LineSolid, CapButt, JoinMiter);
          XDrawLines(p->disp, p->win, p->foregc,
                     ps, p->datacount, 0);
          XSetLineAttributes(p->disp, p->foregc, 0,
                             LineSolid, CapButt, JoinMiter);
        }
        /* 点 */
        else if (!strcmp(p->line_style, GR_LINE_STYLE_DOT)) {
          if (p->dot_size <= 1.0) {
            XDrawPoints(p->disp, p->win, p->foregc,
                        ps, p->datacount, 0);
          }
          else {
            float d = p->dot_size/2;
            int i;
            for (i = 0;i < p->datacount;i++) {
              XFillArc(p->disp, p->win, p->foregc,
                       (int)ps[i].x - d,
                       (int)ps[i].y - d,
                       (int)p->dot_size,
                       (int)p->dot_size,
                       0, 360*64);
            }
          }
        }
        /* 点線 */
        else if (!strcmp(p->line_style, GR_LINE_STYLE_DOTTEDLINE)) {
          XSetLineAttributes(p->disp, p->foregc, p->line_width,
                             LineOnOffDash, CapButt, JoinMiter);
          XSetDashes(p->disp, p->foregc, 0,
                     gr_dash_pattern_dot, GR_DASH_PATTERN_DOT_LEN);
          XDrawLines(p->disp, p->win, p->foregc,
                     ps, p->datacount, 0);
          XSetLineAttributes(p->disp, p->foregc, 0,
                             LineSolid, CapButt, JoinMiter);
        }

        XSetClipMask(p->disp, p->foregc, (int)NULL);
        Free(ps);
    }

    XFlush(p->disp);

    return(0);
}

int
grTextWidth(Display *display, Window window, GC gc,
            char string[], int length,
            unsigned int *textWidth, unsigned int *textHeight)
{
  Font font;
  XFontStruct *font_st;
  XCharStruct char_st;
  XGCValues gcval;
  int font_asc, font_dsc;
  int direction;

  /* 引数チェック */
  if (!string || !*string)
    return(-1);

  /* GC中のフォントの取得 */
  XGetGCValues(display, gc, GCFont | GCForeground | GCBackground, &gcval);

  /* フォント構造体の取得 */
  if ((font_st = XQueryFont(display, gcval.font)) == NULL)
    return(-1);
  XTextExtents(font_st, string, length,
               &direction, &font_asc, &font_dsc, &char_st);

  /* 文字幅の算出(by pixel) */
  *textWidth = XTextWidth(font_st, string, length);

  /* 文字高の算出(by pixel) */
  *textHeight = font_asc + font_dsc;
  return(0);
}

int
grDrawStringDownToUp(Display *display, Window window, GC gc,
                int x, int y, char string[], int length)
{
  Font font;
  XFontStruct *font_st;
  XCharStruct char_st;
  XGCValues gcval;
  int font_asc, font_dsc;
  int direction;
  unsigned int textWidth;
  unsigned int textHeight;
  XImage *src, *dst;
  Pixmap pix;
  int i, j;                             /* カウンタ */
  long pixel;
  unsigned int depth;
  int format;
  int bitmap_pad;

  /* 引数チェック */
```

```
  if (!string || !*string)
    return(-1);

  /* GC中のフォントの取得 */
  XGetGCValues(display, gc, GCFont | GCForeground | GCBackground, &gcval);

  /* フォント構造体の取得 */
  if ((font_st = XQueryFont(display, gcval.font)) == NULL)
    return(-1);
  XTextExtents(font_st, string, length,
               &direction, &font_asc, &font_dsc, &char_st);
  depth = DefaultDepth(display, DefaultScreen(display));

  /* 文字幅の算出(by pixel) */
  textWidth = XTextWidth(font_st, string, length);

  /* 文字高の算出(by pixel) */
  textHeight = font_asc + font_dsc;

  /* 文字列描画用のピクスマップ作成 */
  pix = XCreatePixmap(display, window, textWidth, textHeight, depth);
/*  XSetForeground(display, gc, gcval.background);*/
  XSetForeground(display, gc, WhitePixel(display, 0));
  XSetBackground(display, gc, gcval.foreground);
  XFillRectangle(display, pix, gc, 0, 0, textWidth, textHeight);
  XSetForeground(display, gc, gcval.foreground);
  XSetBackground(display, gc, gcval.background);

  /* 文字列の描画 */
  XDrawString(display, pix, gc, 0, font_asc, string, length);

  /* 文字列描画のピクスマップのコピー */
  format = ZPixmap;
  bitmap_pad = 8;
  src = XGetImage(display, pix, 0, 0, textWidth, textHeight,
                  1, format);
  if (src == NULL) return(-1);

  /* 回転先のイメージ構造体の作成 */
  dst = XCreateImage(display, DefaultVisual(display, DefaultScreen(display)),
                     depth, format, 0, NULL, textHeight, textWidth,
                     bitmap_pad, 0);
  if (dst == NULL) {
    XDestroyImage(src);
    return(-1);
  }
  dst->data = (unsigned char *)malloc(dst->bytes_per_line * textWidth*4);
  if (dst->data == NULL) {
    XDestroyImage(src);
    XFree(dst);
    return(-1);
  }
  /* 文字列の回転 */
  for (i = 0;i < textHeight;i++) {
    for (j = 0;j < textWidth;j++) {
      pixel = XGetPixel(src, j, i);
      XPutPixel(dst, i, textWidth - j - 1, pixel);
    }
  }

  /* ウィンドウへ描画 */
  XPutImage(display, window, gc, dst, 0, 0,
            x - font_asc, y - textWidth, textHeight, textWidth);

  XFreePixmap(display, pix);
  XDestroyImage(src);
  XFree(dst->data);
  return(0);
}

int
grDrawMeter(gr)
int     gr;
{
    GrAttr      *p;
#ifdef GRAPH_DEBUG
    fprintf(stderr, "grDrawMeter(%d)\n", gr);fflush(stderr);
#endif

    if (gr < 0 || gr >= GR_MAX_GRAPH || !gGraph[gr].on)
        return(-1);

    p = gGraph+gr;
    /* チェック */
    if (p->width <= 0 ||
        p->height <= 0 ||
        p->datacount < 0)
        return(-2);

    /* 座標属性の計算 */
    if (p->hold == FALSE) {
        grCalcAxisAttributes(gr);
    }

    /* 描画領域の計算 */
    grCalcAxisOrigin(gr);
    grClearGraph(gr);

    /* 装飾の描画 */
    if (p->hold == FALSE) {
        if (p->decoration_color) {
            grDrawDecoration(gr);
        }
        else {
            GrClear(gr);
        }
    }

    /* ボックス/座標軸の描画 */
#ifdef GRAPH_DEBUG
    fprintf(stderr, "  draw box/axis\n");fflush(stderr);
#endif
    /* y目盛り文字列の描画 */
    if (p->num_ytick_marks > 0) {
        int             i, x1, y1, x, y;
        float           ry, dy, ra, Oa;
        int             Ox, Oy, Oh, L;

        ry = p->yrange[1] - p->yrange[0];
        if (ry == 0) dy = 0;
        else dy = p->Oh/ry;

        Oa = atan((float)p->Oh/p->Ow*2);
        ra = M_PI - 2*Oa;
        Ox = p->Ox + p->Ow/2;
        Oy = p->Oy + p->Oh;
        Oh = p->Oh;

        for (i = 0;i < p->num_ytick_marks;i++) {
            dy = (p->yticks[i]-p->yrange[0])/ry;
            grGetPointByAngle(Ox, Oy, (float)Oa+ra*dy,
                              (float)Oh*GR_POS_METER_TICK_MARKS, &x1, &y1);
```

```
            grGetCenterPos(gr, p->ytick_marks[i], x1, y1, 0, 0, &x, &y);
            XDrawString(p->disp, p->win, p->tick_gc,
                        x, y, p->ytick_marks[i],
                        strlen(p->ytick_marks[i]));
        }
    }

    /* タイトル文字列の描画 */
    if (p->hold == FALSE) {
        if (p->title && *p->title) {
            int x, y;
            unsigned int        w, h;

            w = p->Aw;
            h = p->Ry;
            grGetCenterPos(gr, p->title, p->Ax, p->Ay, w, h, &x, &y);
            XDrawString(p->disp, p->win, p->title_gc,
                        x, y, p->title, strlen(p->title));
        }

        /* xラベル文字列の描画 */
        if (p->xlabel && *p->xlabel) {
        }
    }

    /* x目盛りの描画 */
    if (p->xticks) {
    }

    /* y目盛りの描画 */
    if (p->yticks) {
        XSegment        seg[GR_TICK_MAX_ALL_TICKS];
        int             i, len;
        float           ry, dy, ra, Oa;
        int             Ox, Oy, Oh, L, x, y;
#ifdef GRAPH_DEBUG
        fprintf(stderr, "  draw yticks\n");fflush(stderr);
#endif

        ry = p->yrange[1] - p->yrange[0];
        if (ry == 0) dy = 0;
        else dy = p->Oh/ry;

        Oa = atan((float)p->Oh/p->Ow*2);
        ra = M_PI - 2*Oa;
        Ox = p->Ox + p->Ow/2;
        Oy = p->Oy + p->Oh;
        Oh = p->Oh;

        /* Major目盛りの描画 */
        len = p->Aw*GR_TICK_LENGTH;
        if (len < 2) len = 2;

        for (i = 0;i < p->num_yticks;i++) {
            dy = (p->yticks[i]-p->yrange[0])/ry;
            grGetPointByAngle(Ox, Oy, (float)Oa+ra*dy,
                              (float)Oh*GR_POS_METER_MAJOR_TICKS1, &x, &y);
            seg[i].x1 = x;
            seg[i].y1 = y;
            grGetPointByAngle(Ox, Oy, (float)Oa+ra*dy,
                              (float)Oh*GR_POS_METER_MAJOR_TICKS2, &x, &y);
            seg[i].x2 = x;
            seg[i].y2 = y;
        }
        for (i = 0;i < p->num_yticks;i++) {
            seg[i].x1 -= 1;
            seg[i].x2 -= 1;
        }
        XSetForeground(p->disp, p->decogc, p->decoration_pixel[1]);
        XDrawSegments(p->disp, p->win, p->decogc, seg, p->num_yticks);
        for (i = 0;i < p->num_yticks;i++) {
            seg[i].x1 += 2;
            seg[i].x2 += 2;
        }
        XSetForeground(p->disp, p->decogc, p->decoration_pixel[2]);
        XDrawSegments(p->disp, p->win, p->decogc, seg, p->num_yticks);
        for (i = 0;i < p->num_yticks;i++) {
            seg[i].x1 -= 1;
            seg[i].x2 -= 1;
        }
        XDrawSegments(p->disp, p->win, p->foregc, seg, p->num_yticks);


        /* Minor目盛りの描画 */
        len = p->Aw*GR_TICK_MINOR_LENGTH;
        if (len < 1) len = 1;
        for (i = 0;i < p->num_yticks_minor;i++) {
            dy = (p->yticks_minor[i]-p->yrange[0])/ry;
            grGetPointByAngle(Ox, Oy, (float)Oa+ra*dy,
                              (float)Oh*GR_POS_METER_MINOR_TICKS1, &x, &y);
            seg[i].x1 = x;
            seg[i].y1 = y;
            grGetPointByAngle(Ox, Oy, (float)Oa+ra*dy,
                              (float)Oh*GR_POS_METER_MINOR_TICKS2, &x, &y);
            seg[i].x2 = x;
            seg[i].y2 = y;
        }
        XSetForeground(p->disp, p->decogc, p->decoration_pixel[0]);
        XDrawSegments(p->disp, p->win, p->decogc, seg, p->num_yticks_minor);
        for (i = 0;i < p->num_yticks_minor;i++) {
            seg[i].x1 -= 1;
            seg[i].x2 -= 1;
        }
        XSetForeground(p->disp, p->decogc, p->decoration_pixel[1]);
        XDrawSegments(p->disp, p->win, p->decogc, seg, p->num_yticks_minor);
        for (i = 0;i < p->num_yticks_minor;i++) {
            seg[i].x1 += 2;
            seg[i].x2 += 2;
        }
        XSetForeground(p->disp, p->decogc, p->decoration_pixel[2]);
        XDrawSegments(p->disp, p->win, p->decogc, seg, p->num_yticks_minor);
    }

    /* データの描画 */
    if (p->ydata) {
        XSegment        seg[1];
        int             i, len;
        float           ry, dy, ra, Oa;
        int             Ox, Oy, Oh, L, x, y;
        float           yd;
#ifdef GRAPH_DEBUG
        fprintf(stderr, "  draw ydata\n");fflush(stderr);
#endif
        /* データの平均を求める */
        yd = 0;
        for (i = 0;i < p->datacount;i++) {
            yd += p->ydata[i]/p->datacount;
        }

        ry = p->yrange[1] - p->yrange[0];
        if (ry == 0) dy = 0;
        else dy = p->Oh/ry;
```

```
        Oa = atan((float)p->Oh/p->Ow*2);
        ra = M_PI - 2*Oa;
        Ox = p->Ox + p->Ow/2;
        Oy = p->Oy + p->Oh;
        Oh = p->Oh;

        len = p->Aw*GR_TICK_LENGTH;
        if (len < 2) len = 2;

        dy = (yd-p->yrange[0])/ry;

        /* 範囲チェック */
        if (dy > 1.0)    dy = 1.1;
        else if (dy < 0) dy = -0.1;
        grGetPointByAngle(Ox, Oy, (float)Oa+ra*dy,
                          (float)Oh*GR_POS_METER_STING1, &x, &y);
        seg[0].x1 = x;
        seg[0].y1 = y;
        grGetPointByAngle(Ox, Oy, (float)Oa+ra*dy,
                          (float)Oh*GR_POS_METER_STING2, &x, &y);
        seg[0].x2 = x;
        seg[0].y2 = y;
        XSetForeground(p->disp, p->decogc, p->decoration_pixel[0]);
        XDrawSegments(p->disp, p->win, p->foregc, seg, 1);

        seg[0].x1 -= 1;
        seg[0].x2 -= 1;
        XSetForeground(p->disp, p->decogc, p->decoration_pixel[1]);
        XDrawSegments(p->disp, p->win, p->decogc, seg, 1);

        seg[0].x1 += 2;
        seg[0].x2 += 2;
        XSetForeground(p->disp, p->decogc, p->decoration_pixel[2]);
        XDrawSegments(p->disp, p->win, p->foregc, seg, 1);
    }

    XFlush(p->disp);

    return(0);
}

int
grGetPointByAngle(x, y, a, h, xx, yy)
int     x, y;
float   a;
float   h;
int     *xx, *yy;
{
    float       w;

    w = h/tan(a);
    *xx = x - w;
    *yy = y - h;
    return(0);
}

int
grClearGraph(gr)
int     gr;
{
    GrAttr      *p;
    XPoint      ps[4];
#ifdef GRAPH_DEBUG
    fprintf(stderr, "grClearGraph(%d)\n", gr);fflush(stderr);
#endif

    if (gr < 0 || gr >= GR_MAX_GRAPH || !gGraph[gr].on)
        return(-1);

    p = gGraph+gr;
    ps[0].x = ps[3].x = p->Ox;
    ps[0].y = ps[1].y = p->Oy;
    ps[1].x = ps[2].x = p->Ox+p->Ow;
    ps[2].y = ps[3].y = p->Oy+p->Oh;

    XFillPolygon(p->disp, p->win, p->backgc,
                 ps, 4, Convex, CoordModeOrigin);
    return(0);
}

int
GrClear(gr)
int     gr;
{
    GrAttr      *p;
    XPoint      ps[4];
    unsigned int        width, height;
#ifdef GRAPH_DEBUG
    fprintf(stderr, "GrClear(%d)\n", gr);fflush(stderr);
#endif

    if (gr < 0 || gr >= GR_MAX_GRAPH || !gGraph[gr].on)
        return(-1);

    p = gGraph+gr;
    if (p->width <= 0 ||
        p->height <= 0)
        return(-2);

    grGetWindowSize(gr, &width, &height);
    ps[0].x = ps[3].x = 0;
    ps[0].y = ps[1].y = 0;
    ps[1].x = ps[2].x = width;
    ps[2].y = ps[3].y = height;

    XFillPolygon(p->disp, p->win, p->backgc,
                 ps, 4, Convex, CoordModeOrigin);
    return(0);
}

int
GrClose(gr)
int     gr;
{
    GrAttr      *p;
#ifdef GRAPH_DEBUG
    fprintf(stderr, "GrClose(%d)\n", gr);fflush(stderr);
#endif

    if (gr < 0 || gr >= GR_MAX_GRAPH || gGraph[gr].on)
        return(-1);

    p = gGraph+gr;
    p->on = 0;
    Free(p->xdata);
    Free(p->ydata);
    XFreeGC(p->disp, p->foregc);
    XFreeGC(p->disp, p->backgc);
    Free(p->fontname);
    Free(p->fontbold);
    Free(p->fontitalic);
```

```
    Free(p->title);
    Free(p->xticks);
    Free(p->yticks);
    Free(p->xlabel);
    Free(p->ylabel);
    if (p->xtick_marks) {
        int     i;
        for(i = 0;i < p->num_xtick_marks;i++)
            Free(p->xtick_marks[i]);
        Free(p->xtick_marks);
    }
    if (p->ytick_marks) {
        int     i;
        for(i = 0;i < p->num_ytick_marks;i++)
            Free(p->ytick_marks[i]);
        Free(p->ytick_marks);
    }
    Free(p->xticks_minor);
    Free(p->yticks_minor);
    Free(p->xtick_scale);
    Free(p->ytick_scale);
    Free(p->decoration_color);

    /* タイトルフォントの解放 */
    Free(p->title_fontname);
    Free(p->title_fontbold);
    Free(p->title_fontitalic);
    if (p->title_font)
        XFreeFont(p->disp, p->title_font);
    Free(p->title_color);
    if (p->title_gc)
        XFreeGC(p->disp, p->title_gc);

    /* ラベルフォントの解放 */
    Free(p->label_fontname);
    Free(p->label_fontbold);
    Free(p->label_fontitalic);
    if (p->label_font)
        XFreeFont(p->disp, p->label_font);
    Free(p->label_color);
    if (p->label_gc)
        XFreeGC(p->disp, p->label_gc);

    /* 目盛りフォントの解放 */
    Free(p->tick_fontname);
    Free(p->tick_fontbold);
    Free(p->tick_fontitalic);
    if (p->tick_font)
        XFreeFont(p->disp, p->tick_font);
    Free(p->tick_color);
    if (p->tick_gc)
        XFreeGC(p->disp, p->tick_gc);

    return(0);
}

int
grDrawDecoration(gr)
int     gr;
{
    GrAttr      *p;
    XPoint      ps[6];
    int         pt[4];
    int         W, N;

#ifdef GRAPH_DEBUG
```

```
    fprintf(stderr, "grDrawDecoration(%d)\n", gr);fflush(stderr);
#endif
    p = gGraph+gr;
    if (p->graph_type == GR_TYPE_METER) {
        W = 3;
        N = 3;

        /* 原色 */
        XSetForeground(p->disp, p->decogc, p->decoration_pixel[0]);
        XFillRectangle(p->disp, p->win, p->decogc,
                       p->Ax, p->Ay, p->Aw, p->Ah);

        /* 内枠 */
        pt[0] = pt[1] = W;
        pt[2] = pt[3] = N;
        grDrawFrame(p->disp, p->win, p->decogc,
                    p->decoration_pixel[1],
                    p->decoration_pixel[2],
                    FALSE,
                    p->Rx, p->Ry, p->Rw, p->Rh, pt);

        /* 外枠 */
        N = 1;
        pt[0] = pt[1] = N;
        pt[2] = pt[3] = N;
        grDrawFrame(p->disp, p->win, p->decogc,
                    p->decoration_pixel[1],
                    p->decoration_pixel[2],
                    TRUE,
                    p->Ax, p->Ay, p->Aw, p->Ah, pt);

        XFillRectangle(p->disp, p->win, p->backgc,
                       p->Rx, p->Ry, p->Rw, p->Rh);
        return(0);
    }

    switch (p->decoration_type) {
    case 0:
        break;
    case 1:
        W = 4;
        N = 2;

        /* 原色 */
        XSetForeground(p->disp, p->decogc, p->decoration_pixel[0]);
        XFillRectangle(p->disp, p->win, p->decogc,
                       p->Ax, p->Ay, p->Aw, p->Ah);

        /* 内枠 */
        pt[0] = pt[1] = W;
        pt[2] = pt[3] = N;
        grDrawFrame(p->disp, p->win, p->decogc,
                    p->decoration_pixel[1],
                    p->decoration_pixel[2],
                    FALSE,
                    p->Ox, p->Oy, p->Ow, p->Oh, pt);

        /* 外枠 */
        N = 1;
        pt[0] = pt[1] = N;
        pt[2] = pt[3] = N;
        grDrawFrame(p->disp, p->win, p->decogc,
                    p->decoration_pixel[1],
                    p->decoration_pixel[2],
                    TRUE,
                    p->Ax, p->Ay, p->Aw, p->Ah, pt);
```

```
        XFillRectangle(p->disp, p->win, p->backgc,
                       p->Ox, p->Oy, p->Ow, p->Oh);
        break;

    case 2:
        /* 原色 */
        XSetForeground(p->disp, p->decogc, p->decoration_pixel[0]);
        XFillRectangle(p->disp, p->win, p->decogc,
                       p->Ax, p->Ay, p->Aw, p->Ah);

        /* 内枠 */
        pt[0] = pt[1] = N;
        pt[2] = pt[3] = N;
        grDrawFrame(p->disp, p->win, p->decogc,
                    p->decoration_pixel[1],
                    p->decoration_pixel[2],
                    FALSE,
                    p->Rx, p->Ry, p->Rw, p->Rh, pt);

        XFillRectangle(p->disp, p->win, p->backgc,
                       p->Rx, p->Ry, p->Rw, p->Rh);
        break;
    }
    grClearGraph(gr);

    return(0);
}

int
grDrawFrame(disp, win, gc, high, low, outer, x, y, w, h, pt)
Display *disp;
Window  win;
GC      gc;
unsigned long   high, low;
int     outer;
int     x, y, w, h;
int     pt[4];
{
    XPoint      ps[6];

    if (outer == TRUE) {
        /* 暗色 */
        ps[0].x = x+w; ps[0].y = y+h;
        ps[1].x = 0; ps[1].y = -h;
        ps[2].x = -pt[2]; ps[2].y = pt[3];
        ps[3].x = 0; ps[3].y = h-pt[3]-pt[1];
        ps[4].x = -(w-pt[0]-pt[2]); ps[4].y = 0;
        ps[5].x = -pt[0]; ps[5].y = pt[1];
        XSetForeground(disp, gc, low);
        XFillPolygon(disp, win, gc,
                     ps, 6, Complex, CoordModePrevious);

        /* 明色 */
        ps[0].x = 0; ps[0].y = 0;
        ps[1].x = 0; ps[1].y = h;
        ps[2].x = pt[0]; ps[2].y = -pt[1];
        ps[3].x = 0; ps[3].y = -(h-pt[3]-pt[1]);
        ps[4].x = w-pt[0]-pt[2]; ps[4].y = 0;
        ps[5].x = pt[2]; ps[5].y = -pt[3];
        XSetForeground(disp, gc, high);
        XFillPolygon(disp, win, gc,
                     ps, 6, Complex, CoordModePrevious);
    }
    else {
        /* 暗色 */
        ps[0].x = x; ps[0].y = y;
        ps[1].x = 0; ps[1].y = h;
        ps[2].x = -pt[0]; ps[2].y = pt[1];
        ps[3].x = 0; ps[3].y = -(h+pt[1]+pt[3]);
        ps[4].x = w+pt[0]+pt[2]; ps[4].y = 0;
        ps[5].x = -pt[2]; ps[5].y = pt[3];
        XSetForeground(disp, gc, low);
        XFillPolygon(disp, win, gc,
                     ps, 6, Complex, CoordModePrevious);

        /* 明色 */
        ps[0].x = x+w; ps[0].y = y+h;
        ps[1].x = 0; ps[1].y = -h;
        ps[2].x = pt[2]; ps[2].y = -pt[3];
        ps[3].x = 0; ps[3].y = h+pt[1]+pt[3];
        ps[4].x = -(w+pt[0]+pt[2]); ps[4].y = 0;
        ps[5].x = pt[0]; ps[5].y = -pt[1];
        XSetForeground(disp, gc, high);
        XFillPolygon(disp, win, gc,
                     ps, 6, Complex, CoordModePrevious);
    }
    return(0);
}

int
grMakeTickMarks(ticks, num_ticks, marks, num_marks, tick_scale, max_exp)
float   *ticks;
int     num_ticks;
char    ***marks;
int     *num_marks;
char    **tick_scale;
int     max_exp;
{
    char        **strs;
    int         i, n;
    float       x, x1, x2, ex, exp_val;
    char        str[40];
    char        *fmt;
    char        *p;
#ifdef GRAPH_DEBUG
    fprintf(stderr, "grMakeTickMarks()\n");fflush(stderr);
#endif

    if (num_ticks <= 0) {
        return(0);
    }

    x1 = (ticks[0] > 0 ? ticks[0]: -ticks[0]);
    x2 = (ticks[num_ticks-1] > 0 ? ticks[num_ticks-1]: -ticks[num_ticks-1]);
    x = (x1 < x2 ? x2: x1);
    ex = 1;
    strs = (char **)malloc(sizeof(char *)*num_ticks);

    fmt = "%3.1e";
    sprintf(str, fmt, x);
    p = strrchr(str, 'e');
    if (p) {
        *p++ = '\0';
        n = atoi(p);
        if (max_exp < n || n < -max_exp) {
            ex = exp10(n);
            sprintf(str, "10^%d", n);
            *tick_scale = strdup(str);
        }
#ifdef GRAPH_DEBUG
            fprintf(stderr, "  x1=%g,x2=%g,x=%g,p=%s,n=%d, ex=%g, scale=%s\n",
```

```
                    x1, x2, x, p, n, ex, *tick_scale ? *tick_scale: "null");
#endif
    }

    fmt = "%g";
    for (i = 0;i < num_ticks;i++) {
        sprintf(str, fmt, ticks[i]/ex);
        strs[i] = strdup(str);
    }
    *num_marks = num_ticks;
    *marks = strs;

#ifdef GRAPH_DEBUG
    {
        int     i;
        fprintf(stderr, "  tick marks = [");
        for (i = 0;i < *num_marks;i++)
            fprintf(stderr, "\"%s\" ", (*marks)[i]);
        fprintf(stderr, "]\n");fflush(stderr);
    }
#endif
    return(0);
}

int
grMakeXtickMarks(gr)
int     gr;
{
    GrAttr      *p;
    int         i, x1, y1, x, y;
    float       rx, dx;
    char        str[40];
    char        *fmt;
    double      max_value;
#ifdef GRAPH_DEBUG
    fprintf(stderr, "grMakeXtickMarks(%d)\n", gr);fflush(stderr);
#endif

    if (gr < 0 || gr >= GR_MAX_GRAPH || !gGraph[gr].on)
        return(-1);

    p = gGraph+gr;

    /* x軸の目盛り文字列 */
    if (p->num_xticks > 0 && !p->xticks) {
        fprintf(stderr, "Error: fatal in grMakeXtickMarks(%d)\n",
                p->num_xticks);
        return(-1);
    }
    if (p->num_xticks > 0) {
      max_value = pow(10.0, p->max_exp);
        p->xtick_marks = (char **)malloc(sizeof(char *)*p->num_xticks);
/*      if (p->xrange[0] <= -10000.0 || p->xrange[1] >= 10000.0)*/
        if (p->xrange[0] <= -max_value || p->xrange[1] >= max_value)
            fmt = "%3.1e";
        else
            fmt = "%g";
        for (i = 0;i < p->num_xticks;i++) {
            sprintf(str, fmt, p->xticks[i]);
            p->xtick_marks[i] = strdup(str);
        }
    }
    p->num_xtick_marks = p->num_xticks;

#ifdef GRAPH_DEBUG
    {
        int     i;
        fprintf(stderr, "  xtick_marks = [");
        for (i = 0;i < p->num_xtick_marks;i++)
            fprintf(stderr, "\"%s\" ", p->xtick_marks[i]);
        fprintf(stderr, "]\n");fflush(stderr);
    }
#endif
    return(0);
}

int
grMakeYtickMarks(gr)
int     gr;
{
    GrAttr      *p;
    int         i, x1, y1, x, y;
    float       rx, dx;
    char        str[40];
    char        *fmt;
    double      max_value;

#ifdef GRAPH_DEBUG
    fprintf(stderr, "grMakeYtickMarks(%d)\n", gr);fflush(stderr);
#endif
    if (gr < 0 || gr >= GR_MAX_GRAPH || !gGraph[gr].on)
        return(-1);

    p = gGraph+gr;

    /* y軸の目盛り文字列 */
    if (p->num_yticks > 0 && !p->yticks) {
        fprintf(stderr, "Error: fatal in grMakeYtickMarks (%d)\n",
                p->num_yticks);
        return(-1);
    }
    if (p->num_yticks > 0) {
      max_value = pow(10.0, p->max_exp);
        p->ytick_marks = (char **)malloc(sizeof(char *)*p->num_yticks);
        if (p->yrange[0] <= -max_value || p->yrange[1] >= max_value)
            fmt = "%3.1e";
        else
            fmt = "%g";
        for (i = 0;i < p->num_yticks;i++) {
            sprintf(str, fmt, p->yticks[i]);
            p->ytick_marks[i] = strdup(str);
        }
    }
    p->num_ytick_marks = p->num_yticks;

    return(0);
}

int
grGetWindowSize(gr, width, height)
int             gr;
unsigned int    *width;
unsigned int    *height;
{
    Window              root;
    unsigned int        x, y, border, depth;

    if (gr < 0 || gr >= GR_MAX_GRAPH || !gGraph[gr].on)
        return(-1);

    XGetGeometry(gGraph[gr].disp, gGraph[gr].win, &root,
                 &x, &y, width, height, &border, &depth);
```

```
#ifdef GRAPH_DEBUG
    fprintf(stderr, "grGetWindowSize(%d,%d,%d)\n",
            gr, *width, *height);fflush(stderr);
#endif
    return(0);
}

int
grGetCenterPos(gr, str, x, y, w, h, xx, yy)
int     gr;
char    *str;
int     x, y;
int     w, h;
int     *xx, *yy;
{
    XCharStruct cs;
    int         dir, ascent, decent;
    int width, height;

#ifdef GRAPH_DEBUG
    fprintf(stderr, "grGetCenterPos(%d,\"%s\",%d,%d,%d,%d,...)\n",
            gr, str, x, y, w, h);fflush(stderr);
#endif

    XTextExtents(gGraph[gr].font, str, strlen(str), &dir,
                 &ascent, &decent, &cs);
    width = cs.rbearing - cs.lbearing;
    height = cs.ascent + cs.descent;

    /* 右寄せ */
    if (w < 0)
        *xx = x - (width - cs.lbearing);
    else
        *xx = x + (w - width)/2 + cs.lbearing;
    *yy = y + (h - height)/2 + cs.ascent;
#ifdef GRAPH_DEBUG
    fprintf(stderr, " end of grGetCenterPos()\n");fflush(stderr);
#endif
    return(0);
}

int
grGetCenterPosTitle(gr, str, x, y, w, h, xx, yy)
int     gr;
char    *str;
int     x, y;
int     w, h;
int     *xx, *yy;
{
    XCharStruct cs;
    int         dir, ascent, decent;
    unsigned int        width, height;

#ifdef GRAPH_DEBUG
    fprintf(stderr, "grGetCenterPosByGC(%d,\"%s\",%d,%d,%d,%d,...)\n",
            gr, str, x, y, w, h);fflush(stderr);
#endif
    XTextExtents(gGraph[gr].title_font, str, strlen(str), &dir,
                 &ascent, &decent, &cs);
    width = cs.rbearing - cs.lbearing;
    height = cs.ascent + cs.descent;

    /* 右寄せ */
    if (w < 0)
        *xx = x - (width - cs.lbearing);
    else
        *xx = x + (w - width)/2 + cs.lbearing;
    *yy = y + (h - height)/2 + cs.ascent;
#ifdef GRAPH_DEBUG
    fprintf(stderr, " end of grGetCenterPos()\n");fflush(stderr);
#endif
    return(0);
}

int
grMakeNormalTicks(range, ticks, n, Minor, m)
float   range[2];
float   **ticks;
int     *n;
float   **Minor;
int     *m;
{
    float       w;
    float       tks[GR_TICK_MAX_ALL_TICKS];
    int         i, len;
    int         DIV, N, A, M;
    float       T;
    int         div, c;
    float       t;

#ifdef GRAPH_DEBUG
    fprintf(stderr, "grMakeNormalTicks([%g %g],...)\n",
            range[0], range[1]);fflush(stderr);
#endif
    if (!range || range[0] >= range[1]) {
        *ticks = NULL;
        *n = 0;
        *Minor = NULL;
        *m = 0;
        return(-1);
    }

    /*
     * 値の範囲をGR_TICK_NUM_TICKSで割った商の上1桁の数Aと桁数Nを求める。
     *      A == 1 -> M = 1, c=2
     *      A == 2 -> M = 2, c=2
     *      A <= 5 -> M = 5, c=5
     *      A <= 9 -> M = 10,c=2
     * 上記のMとNより、目盛り間隔Tは
     *      T = M*10^(N-1)
     * で算出する。
     */
    w = (range[1]-range[0])/GR_TICK_NUM_TICKS;
    N = floor(log10(w));
/*    A = floor((double)(w/exp10((double)N)));*/
    A = ceil((double)(w/exp10((double)N)));
    if      (A == 1) M = 1, c = 2;
    else if (A == 2) M = 2, c = 2;
    else if (A <= 5) M = 5, c = 5;
    else             M = 10,c = 2;
    T = M*exp10((double)N);
    DIV = ceil(range[0]/T);

#ifdef GRAPH_DEBUG
    fprintf(stderr, "  w=%g,N=%d, A=%d, M=%d, T=%g, div=%d\n",
            w, N, A, M, T, DIV);
#endif
    for (i = 0;i < GR_TICK_MAX_TICKS;i++) {
        if (range[1] < T*(DIV+i))
            break;
        tks[i] = T*(DIV+i);
    }
```

```
    *n = i;
    len = sizeof(float)*(*n);
    if ((*ticks = (float *)malloc(len)) == NULL) {
      return(-1);
    }
    memcpy(*ticks, tks, len);

    /* Minor目盛りの算出 */
    t = T/c;
    div = ceil(range[0]/t);
    for (i = 0;i < GR_TICK_MAX_ALL_TICKS;i++) {
        if (range[1] < t*(div+i))
            break;
        tks[i] = t*(div+i);
    }
    *m = i;
    len = sizeof(float)*(*m);
    if ((*Minor = (float *)malloc(len)) == NULL) {
      return(-1);
    }
    memcpy(*Minor, tks, len);

    return(0);
}

int
grCalcAxisOrigin(gr)
int     gr;
{
    GrAttr      *p;
    static int          i = 0;
#ifdef GRAPH_DEBUG
    fprintf(stderr, "grCalcAxisOrigin(%d)\n", gr);fflush(stderr);
#endif
    if (gr < 0 || gr >= GR_MAX_GRAPH || !gGraph[gr].on)
        return(-1);

    /* ウィンドウ領域の取得 */
    p = gGraph+gr;

    /*
     * PS出力モードのため以下の関数は使わない
     * grGetWindowSize(gr, &(p->Ww), &(p->Wh));
     */

    /* メータ型のグラフ */
    if (p->graph_type == GR_TYPE_METER) {
        /* 描画領域の算出 */
        p->Ax = 0;
        p->Ay = 0;
        p->Aw = p->Ww;
        p->Ah = p->Wh;

        /* グラフ領域の算出 */
        p->Rx = p->Aw*p->x;
        p->Ry = p->Ah*p->y;
        p->Rw = p->Aw*p->width;
        p->Rh = p->Ah*p->height;

        /* プロット領域の算出 */
        p->Ox = p->Rx+p->Aw*GR_POS_METER_WIDTH_MARGIN/2;
        p->Oy = p->Ry+p->Ah*GR_POS_METER_HEIGHT_MARGIN;
        p->Ow = p->Rw-p->Aw*GR_POS_METER_WIDTH_MARGIN;
        p->Oh = p->Rh-p->Ah*GR_POS_METER_HEIGHT_MARGIN;
    }
```

```
    /* その他通常のグラフ */
    else {
        /* 描画領域の算出 */
        p->Ax = p->x*p->Ww;
        p->Ay = p->y*p->Wh;
        p->Aw = p->width*p->Ww;
        p->Ah = p->height*p->Wh;

        /* グラフ領域の算出 */
        p->Rx = p->Ax + GR_POS_GRAPH_AREA_LEFT_MARGIN;
        p->Ry = p->Ay + GR_POS_GRAPH_AREA_TOP_MARGIN;
        p->Rw = p->Aw - (GR_POS_GRAPH_AREA_LEFT_MARGIN + GR_POS_GRAPH_AREA_RIGHT_MARG
IN);
        p->Rh = p->Ah - (GR_POS_GRAPH_AREA_TOP_MARGIN + GR_POS_GRAPH_AREA_BOTTOM_MARG
IN);

        /* プロット領域の算出 */
        p->Ox = p->Rx + GR_POS_PLOT_AREA_LEFT_MARGIN;
        p->Oy = p->Ry + GR_POS_PLOT_AREA_TOP_MARGIN;
        p->Ow = p->Rw - (GR_POS_PLOT_AREA_LEFT_MARGIN + GR_POS_PLOT_AREA_RIGHT_MARGIN
);
        p->Oh = p->Rh - (GR_POS_PLOT_AREA_TOP_MARGIN + GR_POS_PLOT_AREA_BOTTOM_MARGIN
);

        /* 縦横比が指定されていた場合 */
        if (p->aspect_ratio > 0) {
          float width, height, margin, pitch;

          width  = (float)p->Oh * p->aspect_ratio;
          height = (float)p->Ow / p->aspect_ratio;

          /* 幅を変更する場合 */
          if (width < p->Ow) {
            pitch = (float)p->Ow - width;
            margin = pitch/2.0;
            p->Rx += (int)margin;
            p->Rw -= (int)pitch;
            p->Ax += (int)margin;
            p->Aw -= (int)pitch;
            p->Ox += (int)margin;
            p->Ow -= (int)pitch;
          }
          /* 高さを変更する場合 */
          else if (height < p->Oh) {
            pitch = (float)p->Oh - height;
            margin = pitch/2.0;
            p->Ry += (int)margin;
            p->Rh -= (int)pitch;
            p->Ay += (int)margin;
            p->Ah -= (int)pitch;
            p->Oy += (int)margin;
            p->Oh -= (int)pitch;
          }
        }
    }

    /* グラフ領域の控除 */
/*    if (p->title) {
        p->Oy += (int)(p->Wh*GR_POS_TITLE_HEIGHT);
        p->Oh -= (int)(p->Wh*GR_POS_TITLE_HEIGHT);
    }
    if (p->xlabel) {
        p->Oh -= (int)(p->Wh*GR_POS_XLABEL_HEIGHT);
    }
    if (p->ylabel) {
        p->Ox += (int)(p->Ww*GR_POS_YLABEL_WIDTH);
```

```
        p->Ow -= (int)(p->Ww*GR_POS_YLABEL_WIDTH);
    }*/

    return(0);
}

grGetPixelValueByName(disp, name, color)
Display *disp;
char    *name;
unsigned long *color;
{
    Colormap    cmap = DefaultColormap(disp, 0);
    XColor      c0, c1;

    if (!disp || !name || !*name)
        return(-1);
    XAllocNamedColor(disp, cmap, name, &c1, &c0);
    *color = c1.pixel;
    return(0);
}

int
grAllocHighLowColor(disp, name, pixels)
Display         *disp;
char            *name;
unsigned long   *pixels;
{
    XColor      c;
    Colormap    cmap = DefaultColormap(disp, 0);
    float       h, s, v;
    float       r, g, b;
    float       R = 65535;
    unsigned long plane_mask[8];

#ifdef GRAPH_DEBUG
    fprintf(stderr, "grAllocHighLowColor(%x,%s,)\n", disp,name);fflush(stderr);
#endif
    /* 3つのカラーセルの割り当て */
    if (XAllocColorCells(disp, cmap, FALSE, plane_mask, 0,
                    pixels, 3) == 0) {
        fprintf(stderr, "Error: Can't allocate color cells\n");
        return(-1);
    }

    /* 原色のストア */
    if (XParseColor(disp, cmap, name, &c) == 0) {
        fprintf(stderr, "Error: Can't find color name \"%s\"\n", name);
        return(-2);
    }
    c.pixel = pixels[0];
    XStoreColor(disp, cmap, &c);
    rgb_to_hsv(c.red/R, c.green/R, c.blue/R, &h, &s, &v);
#ifdef GRAPH_DEBUG
    fprintf(stderr, "  source RGB=[%d,%d,%d]\n",
            c.red, c.green, c.blue);
    fprintf(stderr, "  source hsv=[%g,%g,%g]\n",
            h, s, v);fflush(stderr);
#endif

    /* high色のストア */
    hsv_to_rgb(h, s, v+GR_COLOR_HIGH, &r, &g, &b);
    c.red = r*R;
    c.green = g*R;
    c.blue = b*R;
    c.flags = DoRed | DoGreen | DoBlue;
    c.pixel = pixels[1];
    XStoreColor(disp, cmap, &c);
#ifdef GRAPH_DEBUG
    fprintf(stderr, "  high RGB=[%d,%d,%d]\n",
            c.red, c.green, c.blue);fflush(stderr);
#endif

    /* low色のストア */
    hsv_to_rgb(h, s, v+GR_COLOR_LOW, &r, &g, &b);
    c.red = r*R;
    c.green = g*R;
    c.blue = b*R;
    c.flags = DoRed | DoGreen | DoBlue;
    c.pixel = pixels[2];
    XStoreColor(disp, cmap, &c);
#ifdef GRAPH_DEBUG
    fprintf(stderr, "  low RGB=[%d,%d,%d]\n",
            c.red, c.green, c.blue);fflush(stderr);
#endif

    return(0);
}

int
grLoadFontByType(disp, name, bold, size, italic, fs)
Display *disp;
char    *name;
char    *bold;
int     size;
char    *italic;
XFontStruct **fs;
{
    char        buf[BUFSIZ];
    char        it;
    static char *fns[] = GR_FONT_NAME_RANGE, **p;
#ifdef GRAPH_DEBUG
    fprintf(stderr, "grLoadFontByType(,%s,%s,%d,%s,,)\n",
            name ? name: "NULL",
            bold ? bold: "NULL",
            size,
            italic ? italic: "NULL");fflush(stderr);
#endif

    /* フォント名のチェック */
    if (!name)
        name = GR_DEFAULT_FONTNAME;
    else {
        p = fns;
        while (*p) {
            if (!strcmp(name, *p))
                goto next;
            p++;
        }
        fprintf(stderr, "WARNING: Can't find font name \"%s\"\n", name);
        name = GR_DEFAULT_FONTNAME;
    }
 next:

    /* フォントタイプのチェック */
    if (!bold)
        bold = GR_DEFAULT_FONTBOLD;

    /* サイズのチェック */
    if      (size <=  8) size =  8;
    else if (size <= 10) size = 10;
    else if (size <= 12) size = 12;
    else if (size <= 14) size = 14;
```

```
    else if (size <= 18) size = 18;
    else if (size <= 48) size = 24;
    else                 size = GR_DEFAULT_FONTSIZE;

    if (italic) it = italic[0];
    else        it = 'r';

    /* フォント名作成 */
    sprintf(buf, GR_FONT_FORMAT, name, bold, it, size*10);
#ifdef GRAPH_DEBUG
    fprintf(stderr, "  font = \"%s\"\n", buf);fflush(stderr);
#endif
    if (!(*fs = XLoadQueryFont(disp, buf))) {
        return(-1);
    }
    return(0);
}

rgb_to_hsv(red, green, blue, h, s, v)
float red, green, blue;          /* 0〜1.0 で表される rgbの値 */
float *h, *s, *v;                /* 変換後の hsvの値 */
{
        float max, min;
        float r, g, b;

#ifdef GRAPH_DEBUG
    fprintf(stderr, "  RGB={%g,%g,%g}\n",
            red, green, blue);
#endif
        max = red; min = red;

        if (green>max) max = green;
        if (green<min) min = green;

        if (blue>max) max = blue;
        if (blue<min) min = blue;

        *v = max;

        if (max!=0.0 && max != min) *s = (max-min)/max;
        else {
            *s = 0.0;
            *h = 0.0;
            return;
        }

        r = (max - red)/(max - min);
        g = (max - green)/(max - min);
        b = (max - blue)/(max - min);

        if (red == max) *h = b - g;
        else if (green == max) *h = 2 + r - b;
        else if (blue == max) *h = 4 + g - r;

        *h *= 60.0;

        if (*h<0.0) *h += 360.0;
}

hsv_to_rgb(h, s, v, red, green, blue)
     float h, s, v;              /* hsvの値 */
     float *red, *green, *blue; /* 0〜1.0 で表される rgbの値 */
{
        float i, f, p, q, t;

        h /= 60.0;
```

```
        i = (float)((int)h);
        f = h - i;
        p = v*(1-s);
        q = v*(1-s*f);
        t = v*(1 - (s*(1-f)));

        switch ((int)i) {

            case 0:
                *red = v;
                *green = t;
                *blue = p;
                break;

            case 1:
                *red = q;
                *green = v;
                *blue = p;
                break;

            case 2:
                *red = p;
                *green = v;
                *blue = t;
                break;

            case 3:
                *red = p;
                *green = q;
                *blue = v;
                break;

            case 4:
                *red = t;
                *green = p;
                *blue = v;
                break;

            case 5:
                *red = v;
                *green = p;
                *blue = q;
                break;
        }
}

int
grCalcAxisAttributes(gr)
int     gr;
{
    GrAttr      *p;
#ifdef GRAPH_DEBUG
/*    fprintf(stderr, "grCalcAxisAttributes(%d)\n", gr);fflush(stderr);*/
#endif
    if (gr < 0 || gr >= GR_MAX_GRAPH || !gGraph[gr].on) {
        return(-1);
    }

    p = gGraph+gr;
    /* xの最大最小値算出 */
    if (p->xdata && (p->xrange[1] < p->xrange[0])) {
        Number  maxx, minx;
        int     i;

        maxx = -GR_INF;
```

```
        minx = GR_INF;
        for (i = 0;i < p->datacount;i++) {
            if (maxx < p->xdata[i])
                maxx = p->xdata[i];
            if (p->xdata[i] < minx)
                minx = p->xdata[i];
        }
        /* 最大最小値が同じ場合 -1から+1とする */
        if (minx == maxx) {
          minx -= 1;
          maxx += 1;
        }
        p->xrange[0] = minx;
        p->xrange[1] = maxx;
#ifdef GRAPH_DEBUG
        fprintf(stderr, "  xrange(%d) = [%g %g]\n", p->datacount,
                p->xrange[0], p->xrange[1]);
#endif
    }

    /* yの最大最小値算出 */
    if (p->ydata && (p->yrange[1] < p->yrange[0])) {
        Number  maxy, miny;
        int     i;

        maxy = -GR_INF;
        miny = GR_INF;
        for (i = 0;i < p->datacount;i++) {
            if (maxy < p->ydata[i])
                maxy = p->ydata[i];
            if (p->ydata[i] < miny)
                miny = p->ydata[i];
        }
        /* 最大最小値が同じ場合 -1から+1とする */
        if (miny == maxy) {
          miny -= 1;
          maxy += 1;
        }
        p->yrange[0] = miny;
        p->yrange[1] = maxy;
#ifdef GRAPH_DEBUG
        fprintf(stderr, "  yrange=[%g %g]\n",
                p->yrange[0], p->yrange[1]);fflush(stderr);
#endif
    }

    /* x目盛りの計算 */
    if (p->num_xticks != -1) {
#ifdef GRAPH_DEBUG
        fprintf(stderr, "  calc xticks\n");fflush(stderr);
#endif
        grMakeNormalTicks(p->xrange, &(p->xticks), &(p->num_xticks),
                          &(p->xticks_minor), &(p->num_xticks_minor));
        grMakeTickMarks(p->xticks, p->num_xticks,
                        &(p->xtick_marks), &(p->num_xtick_marks),
                        &(p->xtick_scale),
                        p->max_exp);
    }

    /* y目盛りの計算 */
    if (p->num_yticks != -1) {
#ifdef GRAPH_DEBUG
        fprintf(stderr, "  calc yticks\n");fflush(stderr);
#endif
        grMakeNormalTicks(p->yrange, &(p->yticks), &(p->num_yticks),
                          &(p->yticks_minor), &(p->num_yticks_minor));
        grMakeTickMarks(p->yticks, p->num_yticks,
                        &(p->ytick_marks), &(p->num_ytick_marks),
                        &(p->ytick_scale),
                        p->max_exp);
    }

    return(0);
}

int
grFreeAxisAttributes(gr)
int     gr;
{
    GrAttr      *p;
#ifdef GRAPH_DEBUG
/*    fprintf(stderr, "grFreeAxisAttributes(%d)\n", gr);fflush(stderr);*/
#endif
    if (gr < 0 || gr >= GR_MAX_GRAPH || !gGraph[gr].on) {
        return(-1);
    }

    p = gGraph+gr;
    /* x目盛りの開放 */
    Free(p->xticks);
    p->num_xticks = 0;

    /* y目盛りの開放 */
    Free(p->yticks);
    p->num_yticks = 0;

    /* x目盛り文字列の開放 */
    if (p->xtick_marks) {
        int     i;
        for(i = 0;i < p->num_xtick_marks;i++)
            Free(p->xtick_marks[i]);
        Free(p->xtick_marks);
    }
    p->num_xtick_marks = 0;

    /* y目盛り文字列の開放 */
    if (p->ytick_marks) {
        int     i;
        for(i = 0;i < p->num_ytick_marks;i++)
            Free(p->ytick_marks[i]);
        Free(p->ytick_marks);
    }
    p->num_ytick_marks = 0;

    /* x小目盛りの開放 */
    Free(p->xticks_minor);
    p->num_xticks_minor = 0;

    /* y小目盛りの開放 */
    Free(p->yticks_minor);
    p->num_yticks_minor = 0;

    /* x,y範囲の初期化 */
    p->xrange[0] = p->yrange[0] = 0;
    p->xrange[1] = p->yrange[1] = -1;

    return(0);
}

int
grDrawTitleCenter(int gr, Window win, char *title)
{
```

```
  GrAttr *p;
  int x, y;
  unsigned int width, height;
  Window       root;
  unsigned int border, depth;


  p = gGraph + gr;
  XGetGeometry(p->disp, win, &root,
               &x, &y, &width, &height, &border, &depth);
  grGetCenterPosTitle(gr, title,
                 0, 0, width, height, &x, &y);
  XDrawString(p->disp, win, p->title_gc,
              x, y, title,
              strlen(title));
  return(0);
}

int grPSDrawTitleCenter(FILE *fp, int gr, char *title, float *x, float *y,
                        float *width, float *height)
{
  GrAttr *p;

  p = gGraph + gr;
  grPSSetFont(fp,
              p->title_fontname,
              p->title_fontbold,
              p->title_fontitalic,
              p->title_fontsize);
  fprintf(fp, "  (%s) %.5g %.5g %.5g %.5g centerText\n",
          title,
          *x, *y, *width, *height);

  return(0);
}
```

```
/*
 * ACR PLOT TOOL Version 1.0
 *    (c) Copyright 1997
 *    ATR Adaptive Communications Research Laboratories
 *    All Rights Reserved
 */


/*
 * print.c -
 */

#include <stdio.h>
#include <varargs.h>
#include <math.h>
#include <string.h>
#include <X11/X.h>
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <X11/Intrinsic.h>

#include "graph.h"

/*      8       16      24      32      40      48      56                    */
/*------+-------+-------+-------+-------+-------+-------+--------------------*/
/*****************************************************************************
 *        Name: GrPrint
 *    Function: グラフのPS出力
 *    Argument:
 *              Graph gr        I       グラフハンドル
 *              FILE out        I       出力ファイルハンドル
 *      Return:
 *              0  成功
 *             -1  失敗
 * Description:
 *              指定されたファイルへPS形式でグラフの出力を行なう。
 *
 *****************************************************************************/
int GrPrint(Graph gr, FILE *out)
{
    GrAttr      *p;
    int         i;
    XPoint      *ps;

#ifdef GRAPH_DEBUG
    fprintf(stderr, "GrPrint(%d)\n", gr);fflush(stderr);
#endif
    if (gr < 0 || gr >= GR_MAX_GRAPH || !gGraph[gr].on)
        return(-1);

    p = gGraph + gr;
#ifdef GRAPH_DEBUG
    fprintf(stderr, "  hold: %d\n", p->hold);fflush(stderr);
#endif

    /* チェック */
    if (p->width <= 0 ||
        p->height <= 0 ||
        p->datacount < 0)
        return(-2);

    if (p->graph_type == GR_TYPE_METER) {
      /* not implemented */
    }

    /* 座標属性の計算 */
    if (p->hold == FALSE) {
        grCalcAxisAttributes(gr);
    }

    /* 描画領域の計算 */
    grCalcAxisOrigin(gr);
/*    grClearGraph(gr);*/

    /* ボックス/座標軸の描画 */
#ifdef GRAPH_DEBUG
    fprintf(stderr, "  draw box/axis\n");fflush(stderr);
#endif
    if (p->box) {
#ifdef DRAW
      XDrawRectangle(p->disp, p->win, p->foregc,
                     p->Ox, p->Oy, p->Ow, p->Oh);
#else
      fprintf(out, "  %.5g %.5g %.5g %.5g rectangle\n",
              (float)p->Ox/p->Ww,
             -(float)p->Oy/p->Wh,
              (float)(p->Ox + p->Ow)/p->Ww,
             -(float)(p->Oy + p->Oh)/p->Wh);
#endif
    }
    else {
#ifdef DRAW
      /* x座標軸 */
      XDrawLine(p->disp, p->win, p->foregc,
                p->Ox-1, p->Oy-1 + p->Oh+2,
                p->Ox-1 + p->Ow+2, p->Oy-1 + p->Oh+2);
      /* y座標軸 */
      XDrawLine(p->disp, p->win, p->foregc,
                p->Ox-1, p->Oy-1 + p->Oh+2,
                p->Ox-1, p->Oy-1);

      XDrawLine(p->disp, p->win, p->foregc,
                p->Ox-1, p->Oy-1,
                (int)(p->Ox-1 + p->Aw*GR_TICK_LENGTH/3),
                p->Oy-1);
      XDrawLine(p->disp, p->win, p->foregc,
                p->Ox-1 + p->Ow+2, p->Oy-1 + p->Oh+2,
                p->Ox-1 + p->Ow+2,
                (int)((p->Oy-1 + p->Oh+2)-p->Ah*GR_TICK_LENGTH/3));
#else
      /* x座標軸 */
      fprintf(out, "  %.5g %.5g %.5g %.5g line\n",
              (float)p->Ox/p->Ww, (float)-(p->Oy + p->Oh)/p->Wh,
              (float)(p->Ox + p->Ow)/p->Ww, (float)-(p->Oy + p->Oh)/p->Wh);

      /* y座標軸 */
      fprintf(out, "  %.5g %.5g %.5g %.5g line\n",
              (float)(p->Ox)/p->Ww, (float)-(p->Oy + p->Oh)/p->Wh,
              (float)(p->Ox)/p->Ww, (float)-(p->Oy)/p->Wh);

      fprintf(out, "  %.5g %.5g %.5g %.5g line\n",
              (float)p->Ox/p->Ww,
             -(float)p->Oy/p->Wh,
              (float)(p->Ox + p->Aw*GR_TICK_LENGTH/3)/p->Ww,
             -(float)p->Oy/p->Wh);
      fprintf(out, "  %.5g %.5g %.5g %.5g line\n",
              (float)(p->Ox + p->Ow)/p->Ww,
             -(float)(p->Oy + p->Oh)/p->Wh,
              (float)(p->Ox + p->Ow)/p->Ww,
             -(float)((p->Oy + p->Oh)-p->Ah*GR_TICK_LENGTH/3)/p->Wh);
#endif
    }
```

```
    /* x目盛り文字列の描画 */
    if (p->num_xtick_marks > 0) {
      int               i, x1, y1, x, y;
      float     rx, dx;

#ifdef GRAPH_DEBUG
      fprintf(stderr, "  draw xtick marks\n");fflush(stderr);
#endif
#ifdef DRAW
      rx = p->xrange[1] - p->xrange[0];
      if (rx == 0) dx = 0;
      else dx = p->Ow/rx;

      for (i = 0;i < p->num_xtick_marks;i++) {
        x1 = p->Ox + (p->xticks[i]-p->xrange[0])*dx;
        y1 = p->Oy + p->Oh + p->Ah*GR_POS_XTICKS_MARGIN;
        grGetCenterPos(gr, p->xtick_marks[i],
                       x1, y1, 0, GR_POS_XTICKS_HEIGHT, &x, &y);
        XDrawString(p->disp, p->win, p->tick_gc,
                    x, y, p->xtick_marks[i],
                    strlen(p->xtick_marks[i]));
      }
#else
      /* フォントの設定 */
      grPSSetFont(out,
                  p->fontname,
                  p->fontbold,
                  p->fontitalic,
                  p->fontsize);

      rx = p->xrange[1] - p->xrange[0];
      if (rx == 0) dx = 0;
      else dx = p->Ow/rx;
      for (i = 0;i < p->num_xtick_marks;i++) {
        x1 = p->Ox + (p->xticks[i]-p->xrange[0])*dx;
        y1 = p->Oy + p->Oh + p->Ah*GR_POS_XTICKS_MARGIN;
        fprintf(out, "  (%s) %.5g %.5g %.5g %.5g centerText\n",
                p->xtick_marks[i],
                (float)x1/p->Ww,
               -(float)y1/p->Wh,
                (float)0,
               -(float)GR_POS_XTICKS_HEIGHT/p->Wh);
      }
#endif
    }

    /* xスケールの描画 */
    if (p->xtick_scale) {
      int  x, y;
      char str[20], *ptr;
      unsigned int width, height;

#ifdef DRAW
      x = p->Ox + p->Ow;
      y = p->Oy + p->Oh + p->Ah*GR_POS_XTICKS_MARGIN
        + GR_POS_XTICKS_HEIGHT;
      grGetCenterPos(gr, str, x, y, 2, 2, &x, &y);

      strcpy(str, "x");
      strcat(str, p->xtick_scale);
      if ((ptr = strchr(str, '^')) != NULL)
        *ptr++ = '\0';
      XDrawString(p->disp, p->win, p->tick_gc,
                  x, y, str, strlen(str));
      if (ptr) {
        grTextWidth(p->disp, p->win, p->tick_gc,
                    str, strlen(str), &width, &height);
        XDrawString(p->disp, p->win, p->ticksub_gc,
                    x + width, y - height/2, ptr, strlen(ptr));
      }
#else
      x = p->Rx + p->Rw;
      y = p->Oy + p->Oh + p->Ah*GR_POS_XTICKS_MARGIN
        + GR_POS_XTICKS_HEIGHT;

      strcpy(str, "x");
      strcat(str, p->xtick_scale);
      if ((ptr = strchr(str, '^')) != NULL)
        *ptr++ = '\0';

      grPSSetFont(out,
                  p->tick_fontname,
                  p->tick_fontbold,
                  p->tick_fontitalic,
                  p->tick_fontsize);
      fprintf(out, "  (%s) %.5g %.5g rightText\n",
              str,
              (float)x/p->Ww,
             -(float)y/p->Wh);

      if (ptr) {
        grPSSetFont(out,
                    p->ticksub_fontname,
                    p->ticksub_fontbold,
                    p->ticksub_fontitalic,
                    p->ticksub_fontsize);
        fprintf(out, "  (%s) %.5g %.5g leftText\n",
                ptr,
                (float)x/p->Ww,
               -(float)(y-4)/p->Wh);
      }
#endif

    }

    /* y目盛り文字列の描画 */
    if (p->num_ytick_marks > 0) {
      int               i, x1, y1, x, y;
      float             ry, dy;

#ifdef DRAW
      ry = p->yrange[1] - p->yrange[0];
      if (ry == 0) dy = 0;
      else dy = p->Oh/ry;
      for (i = 0;i < p->num_ytick_marks;i++) {
        x1 = p->Ox - GR_POS_YTICKS_MARGIN;
        y1 = p->Oy + p->Oh - (p->yticks[i]-p->yrange[0])*dy;
        grGetCenterPos(gr, p->ytick_marks[i], x1, y1, -1, 0, &x, &y);
        XDrawString(p->disp, p->win, p->tick_gc,
                    x, y, p->ytick_marks[i],
                    strlen(p->ytick_marks[i]));
      }
#else
      /* フォントの設定 */
      grPSSetFont(out,
                  p->fontname,
                  p->fontbold,
                  p->fontitalic,
                  p->fontsize);

      ry = p->yrange[1] - p->yrange[0];
```

```
      if (ry == 0) dy = 0;
      else dy = p->Oh/ry;
      for (i = 0;i < p->num_ytick_marks;i++) {
        x1 = p->Ox - GR_POS_YTICKS_MARGIN;
        y1 = p->Oy + p->Oh - (p->yticks[i]-p->yrange[0])*dy;
        fprintf(out, "  (%s) %.5g %.5g rightText\n",
                p->ytick_marks[i],
                (float)x1/p->Ww,
               -(float)y1/p->Wh);
      }
#endif
    }

    /* タイトル文字列の描画 */
    if (p->title && *p->title) {
      int       x, y;
      unsigned int      w, h;

#ifdef DRAW
      w = p->Aw;
      h = p->Ry;
      grGetCenterPos(gr, p->title, p->Ax, p->Ay, w, h, &x, &y);
      XDrawString(p->disp, p->win, p->title_gc,
                  x, y, p->title, strlen(p->title));
#else
#endif
    }

    /* xラベル文字列の描画 */
    if (p->xlabel && *p->xlabel) {
      int       x, y;
      unsigned int      w, h;

#ifdef DRAW
      w = p->Ow;
      h = p->Ah - (p->Ry + p->Rh);
      grGetCenterPos(gr, p->xlabel, p->Ox, p->Ry+p->Rh, w, h, &x, &y);
      XDrawString(p->disp, p->win, p->label_gc,
                  x, y, p->xlabel, strlen(p->xlabel));
#else
      w = p->Ow;
      h = (p->Ay + p->Ah) - (p->Ry + p->Rh);
      grPSSetFont(out,
                  p->label_fontname,
                  p->label_fontbold,
                  p->label_fontitalic,
                  p->label_fontsize);
      fprintf(out, "  (%s) %.5g %.5g %.5g %.5g centerText\n",
              p->xlabel,
              (float)p->Ox/p->Ww,
             -(float)(p->Ry + p->Rh)/p->Wh,
              (float)w/p->Ww,
             -(float)h/p->Wh);
#endif
    }

    /* yラベル文字列の描画 */
    if (p->ylabel && *p->ylabel) {
      int       x, y;
      unsigned int      w, h;

#ifdef DRAW
      w = p->Rx - p->Ax;
      h = p->Oh;
      grGetCenterPos(gr, p->ylabel, p->Ax, p->Ay, h, w, &y, &x);
      grDrawStringDownToUp(p->disp, p->win, p->label_gc,
```

```
                           x, p->Oy + p->Oh - (y - p->Ay),
                           p->ylabel, strlen(p->ylabel));
#else
      w = p->Oh;
      h = p->Rx - p->Ax;
      grPSSetFont(out,
                  p->label_fontname,
                  p->label_fontbold,
                  p->label_fontitalic,
                  p->label_fontsize);
      fprintf(out, "  (%s) %.5g %.5g %.5g %.5g centerTextRot90\n",
              p->ylabel,
              (float)p->Rx/p->Ww,
             -(float)(p->Oy + p->Oh)/p->Wh,
              (float)w/p->Wh,
             -(float)h/p->Ww);
#endif
    }

    /* yスケールの描画 */
    if (p->ytick_scale) {
      int  x, y;
      char str[20], *ptr;
      unsigned int width, height;

#ifdef DRAW
      x = p->Ox;
      y = p->Ry;
      grGetCenterPos(gr, str, x, y, -1, 0, &x, &y);

      strcpy(str, "x");
      strcat(str, p->ytick_scale);
      if ((ptr = strchr(str, '^')) != NULL)
        *ptr++ = '\0';
      XDrawString(p->disp, p->win, p->tick_gc,
                  x, y, str, strlen(str));
      if (ptr) {
        grTextWidth(p->disp, p->win, p->tick_gc,
                    str, strlen(str), &width, &height);
        XDrawString(p->disp, p->win, p->ticksub_gc,
                    x + width, y - height/2, ptr, strlen(ptr));
      }
#else
      x = p->Ox;
      y = p->Ry;
      strcpy(str, "x");
      strcat(str, p->ytick_scale);
      if ((ptr = strchr(str, '^')) != NULL)
        *ptr++ = '\0';
      grPSSetFont(out,
                  p->tick_fontname,
                  p->tick_fontbold,
                  p->tick_fontitalic,
                  p->tick_fontsize);
      fprintf(out, "  (%s) %.5g %.5g rightText\n",
              str,
              (float)x/p->Ww,
             -(float)y/p->Wh);
      if (ptr) {
        grPSSetFont(out,
                    p->ticksub_fontname,
                    p->ticksub_fontbold,
                    p->ticksub_fontitalic,
                    p->ticksub_fontsize);
        fprintf(out, "  (%s) %.5g %.5g leftText\n",
                ptr,
```

```
                (float)x/p->Ww,
               -(float)(y-4)/p->Wh);
      }

#endif
    }

    /* x目盛りの描画 */
    if (p->xticks) {
#ifdef DRAW
      XSegment  seg[GR_TICK_MAX_ALL_TICKS];
#endif
      int               i, len;
      float     rx, dx;

#ifdef GRAPH_DEBUG
      fprintf(stderr, "  draw xticks\n");fflush(stderr);
#endif
#ifdef DRAW
      rx = p->xrange[1] - p->xrange[0];
      if (rx == 0) dx = 0;
      else dx = p->Ow/rx;

      /* Major目盛りの描画 */
      len = p->Ah*GR_TICK_LENGTH;
      if (len < 2) len = 2;
      for (i = 0;i < p->num_xticks;i++) {
        seg[i].x1 = seg[i].x2 = p->Ox + (p->xticks[i]-p->xrange[0])*dx;
        seg[i].y1 = p->Oy + p->Oh;
        seg[i].y2 = p->Oy + p->Oh - len;
      }
      XDrawSegments(p->disp, p->win, p->foregc, seg, p->num_xticks);

      /* Minor目盛りの描画 */
      len = p->Ah*GR_TICK_MINOR_LENGTH;
      if (len < 1) len = 1;
      if (p->num_xticks_minor > 0) {
        for (i = 0;i < p->num_xticks_minor;i++) {
          seg[i].x1 = seg[i].x2 = p->Ox +
            (p->xticks_minor[i]-p->xrange[0])*dx;
          seg[i].y1 = p->Oy + p->Oh;
          seg[i].y2 = p->Oy + p->Oh - len;
        }
        XDrawSegments(p->disp, p->win, p->foregc,
                      seg, p->num_xticks_minor);
      }
#else
      rx = p->xrange[1] - p->xrange[0];
      if (rx == 0) dx = 0;
      else dx = p->Ow/rx;

      /* Major目盛りの描画 */
      len = p->Ah*GR_TICK_LENGTH;
      if (len < 2) len = 2;
      for (i = 0;i < p->num_xticks;i++) {
        fprintf(out, "  %.5g %.5g %.5g %.5g line\n",
                (float)(p->Ox + (p->xticks[i]-p->xrange[0])*dx)/p->Ww,
                -(float)(p->Oy + p->Oh)/p->Wh,
                (float)(p->Ox + (p->xticks[i]-p->xrange[0])*dx)/p->Ww,
                -(float)(p->Oy + p->Oh - len)/p->Wh);
      }

      /* Minor目盛りの描画 */
      len = p->Ah*GR_TICK_MINOR_LENGTH;
      if (len < 1) len = 1;
      if (p->num_xticks_minor > 0) {
        for (i = 0;i < p->num_xticks_minor;i++) {
          fprintf(out, "  %.5g %.5g %.5g %.5g line\n",
                  (float)(p->Ox + (p->xticks_minor[i]-p->xrange[0])*dx)/p->Ww,
                  -(float)(p->Oy + p->Oh)/p->Wh,
                  (float)(p->Ox + (p->xticks_minor[i]-p->xrange[0])*dx)/p->Ww,
                  -(float)(p->Oy + p->Oh - len)/p->Wh);
        }
      }
#endif

    }

    /* y目盛りの描画 */
    if (p->yticks) {
#ifdef DRAW
      XSegment  seg[GR_TICK_MAX_ALL_TICKS];
#endif
      int               i, len;
      float             ry, dy;
#ifdef GRAPH_DEBUG
      fprintf(stderr, "  draw yticks\n");fflush(stderr);
#endif

#ifdef DRAW
      ry = p->yrange[1] - p->yrange[0];
      if (ry == 0) dy = 0;
      else dy = p->Oh/ry;

      /* Major目盛りの描画 */
      len = p->Aw*GR_TICK_LENGTH;
      if (len < 2) len = 2;
      for (i = 0;i < p->num_yticks;i++) {
        seg[i].y1 = seg[i].y2 = p->Oy + p->Oh
          - (p->yticks[i]-p->yrange[0])*dy;
        seg[i].x1 = p->Ox;
        seg[i].x2 = p->Ox + len;
      }
      XDrawSegments(p->disp, p->win, p->foregc, seg, p->num_yticks);

      /* Minor目盛りの描画 */
      len = p->Aw*GR_TICK_MINOR_LENGTH;
      if (len < 1) len = 1;
      for (i = 0;i < p->num_yticks_minor;i++) {
        seg[i].y1 = seg[i].y2 = p->Oy + p->Oh
          - (p->yticks_minor[i]-p->yrange[0])*dy;
        seg[i].x1 = p->Ox;
        seg[i].x2 = p->Ox + len;
      }
      XDrawSegments(p->disp, p->win, p->foregc, seg,
                    p->num_yticks_minor);
#else
      ry = p->yrange[1] - p->yrange[0];
      if (ry == 0) dy = 0;
      else dy = p->Oh/ry;

      /* Major目盛りの描画 */
      len = p->Aw*GR_TICK_LENGTH;
      if (len < 2) len = 2;
      for (i = 0;i < p->num_yticks;i++) {
        fprintf(out, "  %.5g %.5g %.5g %.5g line\n",
                (float)(p->Ox)/p->Ww,
                -(float)(p->Oy + p->Oh - (p->yticks[i]-p->yrange[0])*dy)/p->Wh,
                (float)(p->Ox + len)/p->Ww,
                -(float)(p->Oy + p->Oh - (p->yticks[i]-p->yrange[0])*dy)/p->Wh);
      }
```

```
      /* Minor目盛りの描画 */
      len = p->Aw*GR_TICK_MINOR_LENGTH;
      if (len < 1) len = 1;
      if (p->num_yticks_minor > 0) {
        for (i = 0;i < p->num_yticks_minor;i++) {
          fprintf(out, "  %.5g %.5g %.5g %.5g line\n",
                  (float)(p->Ox)/p->Ww,
                 -(float)(p->Oy + p->Oh
                          - (p->yticks_minor[i]-p->yrange[0])*dy)/p->Wh,
                  (float)(p->Ox + len)/p->Ww,
                 -(float)(p->Oy + p->Oh
                          - (p->yticks_minor[i]-p->yrange[0])*dy)/p->Wh);
        }
      }
#endif
    }

    /* データの描画 */
    if (p->ydata) {
      XRectangle        rect[1];
      Number          x, y, oldx, oldy;

      /* データの相対座標から絶対座標への変換 */
      ps = (XPoint *)malloc(sizeof(XPoint)*(p->datacount));
#ifdef DRAW
      for (i = 0;i < p->datacount;i++) {
        x = (p->Ox + (p->xdata[i]-p->xrange[0])
             /(p->xrange[1]-p->xrange[0])*p->Ow);
        y = (p->Oy + p->Oh - (p->ydata[i]-p->yrange[0])
             /(p->yrange[1]-p->yrange[0])*p->Oh);
        ps[i].x = rint(x);
        ps[i].y = rint(y);
      }
#endif
#ifdef GRAPH_DEBUG_EX
      {
        int             i;
        fprintf(stderr, "  xrange = [%g %g], yrange = [%g %g]\n",
                p->xrange[0], p->xrange[1],
                p->yrange[0], p->yrange[1]);
        fprintf(stderr, "  data = [");
        for (i = 0;i < p->datacount;i++)
          fprintf(stderr, "(%g,%g;%d,%d)",
                  p->xdata[i], p->ydata[i],
                  ps[i].x, ps[i].y);
        fprintf(stderr, "]\n");
      }
#endif
      /* プロット領域をクリップする */
      rect[0].x = p->Ox-1;
      rect[0].y = p->Oy-1;
      rect[0].width = p->Ow+2;
      rect[0].height = p->Oh+2;
#ifdef DRAW
      XSetClipRectangles(p->disp, p->foregc,
                         0, 0, rect, 1, Unsorted);
#endif

      fprintf(out, "  %.5g %.5g %.5g %.5g setAxesScale\n",
              (float)p->Ox/p->Ww, -(float)(p->Oy + p->Oh)/p->Wh,
              (float)p->Ow/p->Ww, (float)p->Oh/p->Wh);
      fprintf(out, "  %.5g setlinewidth\n", p->line_width/10.0);
      /*
       * 線種による描きわけ
       */
      /* 直線 */
      if (!strcmp(p->line_style, GR_LINE_STYLE_LINE)) {
#ifdef DRAW
        XSetLineAttributes(p->disp, p->foregc, p->line_width,
                           LineSolid, CapButt, JoinMiter);
        XDrawLines(p->disp, p->win, p->foregc,
                   ps, p->datacount, 0);
        XSetLineAttributes(p->disp, p->foregc, 0,
                           LineSolid, CapButt, JoinMiter);
#endif

        oldx = (p->xdata[0]-p->xrange[0])
          /(p->xrange[1]-p->xrange[0]);
        oldy = (p->ydata[0]-p->yrange[0])
          /(p->yrange[1]-p->yrange[0]);
        for (i = 1;i < p->datacount;i++) {
          x = (p->xdata[i]-p->xrange[0])
            /(p->xrange[1]-p->xrange[0]);
          y = (p->ydata[i]-p->yrange[0])
            /(p->yrange[1]-p->yrange[0]);
          fprintf(out, "  %.5g %.5g %.5g %.5g line\n", oldx, oldy, x, y);
          oldx = x;
          oldy = y;
        }
      }
      /* 点 */
      else if (!strcmp(p->line_style, GR_LINE_STYLE_DOT)) {
#ifdef DRAW
        XDrawPoints(p->disp, p->win, p->foregc,
                    ps, p->datacount, 0);
#endif

        x = (p->xdata[0]-p->xrange[0])
          /(p->xrange[1]-p->xrange[0]);
        y = (p->ydata[0]-p->yrange[0])
          /(p->yrange[1]-p->yrange[0]);
        fprintf(out, "  %.6g %.6g dot1\n", x, y);
        for (i = 1;i < p->datacount;i++) {
          x = (p->xdata[i]-p->xrange[0])
            /(p->xrange[1]-p->xrange[0]);
          y = (p->ydata[i]-p->yrange[0])
            /(p->yrange[1]-p->yrange[0]);
          fprintf(out, "  %.5g %.5g dot1\n", x, y);
        }
      }
/* 点線 */
      else if (!strcmp(p->line_style, GR_LINE_STYLE_DOTTEDLINE)) {
#ifdef DRAW
        XSetLineAttributes(p->disp, p->foregc, p->line_width,
                           LineOnOffDash, CapButt, JoinMiter);
        XSetDashes(p->disp, p->foregc, 0,
                   gr_dash_pattern_dot, GR_DASH_PATTERN_DOT_LEN);
        XDrawLines(p->disp, p->win, p->foregc,
                   ps, p->datacount, 0);
        XSetLineAttributes(p->disp, p->foregc, 0,
                           LineSolid, CapButt, JoinMiter);
#endif
      }

      fprintf(out, "  stroke\n");
      fprintf(out, "  0.5 setlinewidth\n");
      fprintf(out, "  %.5g %.5g %.5g %.5g restoreAxesScale\n",
              (float)p->Ox/p->Ww, -(float)(p->Oy + p->Oh)/p->Wh,
              (float)p->Ow/p->Ww, (float)p->Oh/p->Wh);
#ifdef DRAW
      XSetClipMask(p->disp, p->foregc, (int)NULL);
#endif
```

```
      Free(ps);
    }

    return(0);
}

int grPSSetFont(FILE *fp, char *fontName, char *fontBold,
                char *fontItalic, int fontSize)
{
  char name[40];
  char bold[40];
  char italic[40];

  if (!fontName)
    return(-1);

  strcpy(name, fontName);
  *name = (char)toupper(*name);
  if (fontBold && strcmp(fontBold, "medium")) {
    strcpy(bold, fontBold);
    *bold = (char)toupper(*bold);
    strcat(name, "-");
    strcat(name, bold);
  }
  else if (fontItalic && strcmp(fontItalic, "roman")) {
    strcpy(bold, fontItalic);
    *italic = (char)toupper(*italic);
    strcat(name, "-");
    strcat(name, italic);
  }
  fprintf(fp, "/%s %d setFontInfo\n", name, fontSize);
  return(0);
}

int PSDrawLines(Number xdata[], Number ydata[], int datacount,
                Number xrange[], Number yrange[])
{
  int i;
  Number dx, dy;

  dx = xrange[1] - xrange[0];
  dy = yrange[1] - yrange[0];

  for (i = 0;i < datacount;i++) {
    printf("  %lg %lg line\n",
           (xdata[i]-xrange[0])/dx,
           (ydata[i]-yrange[0])/dy);
  }
  return(0);
}
```

```
/*
 * ACR PLOT TOOL Version 1.0
 *    (c) Copyright 1997
 *    ATR Adaptive Communications Research Laboratories
 *    All Rights Reserved
 */


/*
 * read.c - データファイル読み込み
 */

#include <stdio.h>

#include "plot.h"

static int gLineNo;                                /* データファイルの行番号 */

/*
 * データファイルの読み込み
 */
DATA *ReadDataFile(char *filename)
{
  FILE *fp;
  char *buf, *p;
  DATA *data;
  int i, j;
  double val;
#ifdef DEBUG
  fprintf(stderr, "ReadDataFile(%s)\n", filename);
#endif

  /* データファイルのオープン */
  if ((fp = fopen(filename, "r")) == NULL) {
    fprintf(stderr, "Cannot open file '%s'.\n", filename);
    return(NULL);
  }

  /* データの領域確保 */
  if ((data = (DATA *)malloc(sizeof(DATA))) == NULL) {
    fprintf(stderr, "Cannot allocate memory.\n");
    fclose(fp);
    return(NULL);
  }

  /* タイトル文字列の読み込み */
  if ((buf = ReadLine(fp)) == NULL) {
    fprintf(stderr, "%s:%d: unexpected end of file\n",
            filename, gLineNo);
    return(NULL);
  }

#ifdef DEBUG
  fprintf(stderr, "[%s]", buf);
#endif
/*  if (strncmp(buf, HEADER_TITLE, strlen(HEADER_TITLE))) {
    fprintf(stderr, "Not found title string.\n");
    return(NULL);
  }*/
/*  strcpy(data->title, buf+strlen(HEADER_TITLE));*/
  strcpy(data->title, buf);

  /* グラフの数、データの行数、データ列数の読み込み */
  if ((buf = ReadLine(fp)) == NULL) return(NULL);
  sscanf(buf, "%d %d %d",
         &(data->nGraphs), &(data->nRows), &(data->nCols));
```

```
#ifdef DEBUG
  fprintf(stderr, "nGraphs=%d,nRows=%d,nCols=%d\n",
          data->nGraphs, data->nRows, data->nCols);
#endif

  /* チェック */
  if (data->nGraphs <= 0 || data->nGraphs > MAX_GRAPH_COUNT) {
    fprintf(stderr, "%s:%d: number of graphs is out of range (%d)\n",
            filename, gLineNo, data->nGraphs);
    fclose(fp);
    return(NULL);
  }
  if (data->nCols <= 0) {
    fprintf(stderr, "%s:%d: number of columns is less than 0 (%d)\n",
            filename, gLineNo, data->nCols);
    fclose(fp);
    return(NULL);
  }
  if (data->nRows <= 0) {
    fprintf(stderr, "%s:%d: number of rows is less than 0 (%d)\n",
            filename, gLineNo, data->nRows);
    fclose(fp);
    return(NULL);
  }

  /* グラフのX/Y軸の列番号、線種の読み込み */
  for (i = 0;i < data->nGraphs;i++) {
    if ((buf = ReadLine(fp)) == NULL) return(NULL);
    sscanf(buf, "%d %d %d",
           &(data->colX[i]), &(data->colY[i]), &(data->lineType[i]));

    /* チェック */
    if (data->colX[i] < 1 || data->colX[i] > data->nCols) {
      fprintf(stderr, "%s:%d: number of x column no. is out of range (%d)\n",
              filename, gLineNo, data->colX[i]);
      fclose(fp);
      return(NULL);
    }
    if (data->colY[i] < 1 || data->colY[i] > data->nCols) {
      fprintf(stderr, "%s:%d: number of y column no. is out of range (%d)\n",
              filename, gLineNo, data->colY[i]);
      fclose(fp);
      return(NULL);
    }
  }

  /* グラフのX/Y軸のラベルの取得 */
  for (i = 0;i < data->nCols;i++) {
    if ((buf = ReadLine(fp)) == NULL) {
      fprintf(stderr, "%s:%d: unexpected end of file\n",
              filename, gLineNo);
      return(NULL);
    }
    strncpy(data->colHeader[i], buf, MAX_HEADER_LEN);
#ifdef DEBUG
    fprintf(stderr, "label %d: %s\n", i, data->colHeader[i]);
#endif
  }

  /* データ領域の確保 */
  if ((data->table = (Number *)malloc(sizeof(Number)*(data->nCols*data->nRows))) == N
ULL) {
    fprintf(stderr, "Cannot allocate memory.(%dx%d)\n",
            data->nCols, data->nRows);
    fclose(fp);
    return(NULL);
```

```
  }

  /* データの読み込み */
  for (i = 0;i < data->nRows;i++) {

    /* データを一行読み込む */
    if ((buf = ReadLine(fp)) == NULL) {
      fprintf(stderr, "%s:%d: Warning: valid rows less than previous.(%d<%d)\n",
              filename, gLineNo, i, data->nRows);
      break;
    }
    if (*buf == '@') {
      break;
    }

    p = buf;
    for (j = 0;j < data->nCols;j++) {
      if (*p == '\0') {
        fprintf(stderr, "%s:%d: valid columns less than previous.(%d<%d)\n",
                filename, gLineNo, j, data->nCols);
        fclose(fp);
        return(NULL);
      }
      sscanf(p, "%lf", &val);

      /* 列優先でデータを格納する */
      *(data->table + j*data->nRows + i) = (Number)val;
#ifdef DEBUG
      fprintf(stderr, "%s(%lf),", p, val);
#endif

      /* 次の空白文字までよみ飛ばす */
      while (*p != ' ' && *p != '\t' && *p != '\n' && *p != '\0') p++;

      /* 次の空白以外の文字までよみ飛ばす */
      while (*p == ' ' || *p == '\t' || *p == '\n') p++;

    }
  }
  data->nValidRows = i;

  fclose(fp);
  return(data);
}

/*
 * ファイルから一行よみ出す
 *     空行をよみ飛ばすことに注意
 */
char *ReadLine(FILE *fp)
{
  static char buf[MAX_BUFFER_LEN+1];
  char *p;

/*  gLineNo = 0;*/
  while (fgets(buf, MAX_BUFFER_LEN, fp)) {
    gLineNo++;
    p = buf;
    while (*p == ' ' || *p == '\t' || *p == '\n') p++;
    if (*p != '\0') {
      if (p[strlen(p)-1] == '\n')
        p[strlen(p)-1] = '\0';
      return(p);
    }
  }
```

```
  fclose(fp);
  return(NULL);
}
```

```
/*
 * ACR PLOT TOOL Version 1.0
 *     (c) Copyright 1997
 *     ATR Adaptive Communications Research Laboratories
 *     All Rights Reserved
 */


/*
 * plot.h - プロット情報定義ファイル
 */
#ifndef __plot_h__
#define __plot_h__
#include <stdio.h>
#include <X11/Xlib.h>

#define Number                  float           /* 数値演算の精度 */

/* 用紙方向の定数 */
#define PAPER_LANDSCAPE         0               /* 横置き */
#define PAPER_PORTRAIT          1               /* 縦置き */

/* モード */
#define X_DRAW_MODE             0               /* X描画モード */
#define PS_ONLY_MODE            1               /* PS出力モード */

#define MAX_TITLE_LEN           256             /* タイトルの最大長さ */
#define MAX_HEADER_LEN          256             /* ヘッダの最大長さ */
#define MAX_GRAPH_COUNT         12              /* グラフの最大数 */
#define MAX_COLUMN_COUNT        100             /* 列の最大数 */
#define STOP_CHAR               '@'             /* 読み込み停止文字 */
#define COMMENT_CHAR            '!'             /* コメント文字 */
#define MAX_BUFFER_LEN          4096            /* 行バッファの最大長さ */
#define HEADER_TITLE            "タイトル："
#define ulong                   unsigned long   /* 正長整数 */

/* デフォルト値の定義 */
#define DEFAULT_FILENAME        "fort.99"
#define DEFAULT_MAIN_WINDOW_WIDTH  (290*3)      /* 初期ウィンドウの幅 */
#define DEFAULT_MAIN_WINDOW_HEIGHT (200*3)      /* 初期ウィンドウの高さ */
#define DEFAULT_TITLE_WINDOW_HEIGHT 0.06        /* タイトルウィンドウの大きさ(相対値) */
#define DEFAULT_PAPER_ORIENTATION (PAPER_LANDSCAPE) /* 用紙の向き */

/* プロット情報の定義 */
typedef struct _DATA {
  Display *display;                     /* ディスプレイ */
  char title[MAX_TITLE_LEN+1];          /* タイトル */
  int nGraphs;                          /* グラフの数 */
  int nRows;                            /* データ行数 */
  int nCols;                            /* データ列数 */
  int nValidRows;                       /* 有効列数 */
  int colX[MAX_GRAPH_COUNT];            /* X軸の列番号 */
  int colY[MAX_GRAPH_COUNT];            /* Y軸の列番号 */
  Number rangeX[MAX_GRAPH_COUNT][2];    /* X軸の範囲 */
  Number rangeY[MAX_GRAPH_COUNT][2];    /* Y軸の範囲 */
  int lineType[MAX_GRAPH_COUNT];        /* グラフの線種 */
  char colHeader[MAX_COLUMN_COUNT][MAX_HEADER_LEN+1];  /* グラフのヘッダ */
  Window winId[MAX_GRAPH_COUNT];        /* ウィンドウのID */
  int winX[MAX_GRAPH_COUNT];            /* ウィンドウの左上のx座標 */
  int winY[MAX_GRAPH_COUNT];            /* ウィンドウの左上のy座標 */
  unsigned int winWidth[MAX_GRAPH_COUNT];  /* 各グラフウィンドウの幅 */
  unsigned int winHeight[MAX_GRAPH_COUNT]; /* 各グラフウィンドウの高 */
  Number *table;                        /* データの配列 */
  Window mainWinId;                     /* メインウィンドウのID */
  unsigned int mainWinWidth;            /* メインウィンドウの幅 */
  unsigned int mainWinHeight;           /* メインウィンドウの高 */
  int orientation;                      /* 用紙／画面の方向 */
  Window titleWinId;                    /* タイトルウィンドウのID */
  int titleWinX;                        /* タイトルウィンドウのx座標 */
  int titleWinY;                        /* タイトルウィンドウのy座標 */
  unsigned int titleWinWidth;           /* タイトルウィンドウの幅 */
  unsigned int titleWinHeight;          /* タイトルウィンドウの高 */
  int mode;                             /* 描画/出力モード */
  float aspectRatio;                    /* 縦横比 */
  float dotSize;                        /* 点の大きさ */
} DATA;

#if 0
/* プロット情報の初期化 */
#define INIT_DATA(d) \
    {   int i; \
      (d)->title[0] = '\0'; \
      (d)->nGraphs = 0; \
      (d)->nRows = 0; \
      (d)->nCols = 0; \
      (d)->nValidRows = 0; \
      for (i = 0;i < MAX_GRAPH_COUNT;i++) { \
        (d)->colX[i] = 0; \
        (d)->colY[i] = 0; \
        (d)->rangeX[i][0] = 0; \
        (d)->rangeX[i][1] = 1; \
        (d)->rangeY[i][0] = 0; \
        (d)->rangeY[i][1] = 1; \
        (d)->lineType[i] = 0; \
        (d)->winId[i] = 0; \
        (d)->winX[i] = 0; \
        (d)->winY[i] = 0; \
        (d)->winWidth[i] = 1; \
        (d)->winHeight[i] = 1; \
      } \
      for (i = 0;i < MAX_COLUMN_COUNT;i++) { \
        (d)->colHeader[i][0] = '\0'; \
      } \
      (d)->table = NULL; \
      (d)->mainWinId = 0; \
      (d)->mainWinWidth = 1; \
      (d)->mainWinHeight = 1; \
      (d)->orientation = DEFAULT_PAPER_ORIENTATION; \
      (d)->titleWinId = 0; \
      (d)->titleWinX = 1; \
      (d)->titleWinY = 1; \
      (d)->titleWinWidth = 1; \
      (d)->titleWinHeight = 1; \
      (d)->mode = X_DRAW_MODE; \
      (d)->aspectRatio = -1; \
      (d)->dotSize = 0.8; \
    }
#endif

DATA *ReadDataFile(char *);
char *ReadLine(FILE *);

#endif  /* end of __plot_h__ */
```

```
/*
 * ACR PLOT TOOL Version 1.0
 *     (c) Copyright 1997
 *     ATR Adaptive Communications Research Laboratories
 *     All Rights Reserved
 */


/*
 * graph.h - グラフ情報の定義ファイル
 */

#ifndef __graph_h__
#define __graph_h__

#define Number  float
#define Graph                   int             /* グラフハンドル */

#define GR_MAX_GRAPH            100
#define GR_INF                  1000000
#define GR_EPS                  1.0e-6
#define GR_DEFFONTNAME  "-adobe-helvetica-bold-r-normal--0-0-75-75-p-0-iso8859-1"

#define GR_INT                  1
#define GR_FLOAT                2
#define GR_CHAR                 3
#define GR_STRING               4
#define GR_ADDR                 5
#define GR_DOUBLE               6
#define GR_SHORT                7

#define GR_TYPE_XY              1
#define GR_TYPE_METER           2

#define GR_X                            1
#define GR_Y                            2
#define GR_WIDTH                        3
#define GR_HEIGHT                       4
#define GR_XDATA                        5
#define GR_XDATATYPE                    6
#define GR_YDATA                        7
#define GR_DATACOUNT                    8
#define GR_YDATATYPE                    9
#define GR_TITLE                        10
#define GR_WINDOW                       11
#define GR_BOX                          12
#define GR_XTICKS                       13
#define GR_NUM_XTICKS                   14
#define GR_YTICKS                       15
#define GR_NUM_YTICKS                   16
#define GR_XAXIS                        17
#define GR_YAXIS                        18
#define GR_XLABEL                       19
#define GR_YLABEL                       20
#define GR_DECORATION_COLOR             21
#define GR_XRANGE                       22
#define GR_YRANGE                       23
#define GR_FONTNAME                     24
#define GR_FONTBOLD                     25
#define GR_FONTITALIC                   26
#define GR_FONTSIZE                     27
#define GR_TITLE_FONTNAME               28
#define GR_TITLE_FONTBOLD               29
#define GR_TITLE_FONTITALIC             30
#define GR_TITLE_FONTSIZE               31
#define GR_TITLE_COLOR                  32
```

```
#define GR_LABEL_FONTNAME               33
#define GR_LABEL_FONTBOLD               34
#define GR_LABEL_FONTITALIC             35
#define GR_LABEL_FONTSIZE               36
#define GR_LABEL_COLOR                  37
#define GR_DRAW_MODE                    38
#define GR_DRAW_SIZE                    39
#define GR_HOLD                         40
#define GR_MAX_EXP                      41
#define GR_GRAPH_TYPE                   42
#define GR_DECORATION_TYPE              43
#define GR_LINE_STYLE                   44
#define GR_TICK_FONTNAME                45  /* 目盛りフォント名 */
#define GR_TICK_FONTBOLD                46  /* 目盛りフォント太さ */
#define GR_TICK_FONTITALIC              47  /* 目盛りフォント傾斜 */
#define GR_TICK_FONTSIZE                48  /* 目盛りフォントサイズ */
#define GR_TICK_COLOR                   49  /* 目盛りフォント色 */
#define GR_TICKSUB_FONTNAME             50  /* 目盛りフォント名 */
#define GR_TICKSUB_FONTBOLD             51  /* 目盛りフォント太さ */
#define GR_TICKSUB_FONTITALIC           52  /* 目盛りフォント傾斜 */
#define GR_TICKSUB_FONTSIZE             53  /* 目盛りフォントサイズ */
#define GR_TICKSUB_COLOR                54  /* 目盛りフォント色 */
#define GR_LINE_WIDTH                   55
#define GR_ASPECT_RATIO                 56
#define GR_DOT_SIZE                     57

#define GR_LINE_STYLE_LINE                "-"
#define GR_LINE_STYLE_DOT                 "."
#define GR_LINE_STYLE_DOTTEDLINE          "-."
#define GR_LINE_STYLE_DASH                "--"
#define GR_LINE_STYLE_ROUND               "o"
#define GR_LINE_STYLE_X                   "x"

/*
 * グラフ情報宣言
 */
typedef struct {
    int         on;                     /* 有効フラグ */
    Display     *disp;                  /* ディスプレイ構造体 */
    Window      win;                    /* Xウィンドウ ID */
    float       x;                      /* 左上相対座標 */
    float       y;                      /* 左上相対座標 */
    float       width;                  /* 相対幅 */
    float       height;                 /* 相対高さ */
    char        line_style[3];          /* 線種 */
    float       line_width;             /* 線幅 */
    float       dot_size;
    Number      *xdata;                 /* xデータ */
    int         xdatatype;              /*  */
    Number      xrange[2];              /* xデータ範囲 */
    Number      *ydata;                 /* yデータ配列 */
    int         ydatatype;              /*  */
    int         datacount;              /* データ数 */
    float       yrange[2];              /* yデータ範囲 */
    char        *title;                 /* タイトル文字列 */
    int         box;                    /* ボックスフラグ */
    float       aspect_ratio;           /* 縦横比 */
    GC          foregc;                 /* 前景GC */
    GC          backgc;                 /* 後景GC */
    char        *fontname;              /* フォント名 */
    char        *fontbold;              /* フォント太さ */
    char        *fontitalic;            /* フォント傾き */
    int         fontsize;               /* フォントサイズ */
    float       *xticks;                /* x目盛り配列 */
    int         num_xticks;             /* x目盛り配列数 */
    float       *yticks;                /* y目盛り配列 */
```

```
    int         num_yticks;                 /* y目盛り配列数 */
    int         xaxis;                      /*  */
    int         yaxis;                      /*  */
    char        *xlabel;                    /* xラベル */
    char        *ylabel;                    /* yラベル */
    char        **xtick_marks;              /* x目盛り文字列 */
    int         num_xtick_marks;            /* x目盛り文字列数 */
    char        **ytick_marks;              /* y目盛り文字列 */
    int         num_ytick_marks;            /* y目盛り文字列数 */
    float       *xticks_minor;              /* x Minor目盛り配列 */
    int         num_xticks_minor;           /* x Minor目盛り配列数 */
    float       *yticks_minor;              /* y Minor目盛り配列 */
    int         num_yticks_minor;           /* y Minor目盛り配列数 */
    char        *xtick_scale;               /* x スケール文字列 */
    char        *ytick_scale;               /* y スケール文字列 */
    int         decoration_type;            /* 装飾タイプ */
    char        *decoration_color;          /* 装飾色 */
    unsigned long decoration_pixel[3];      /* 装飾ピクセル値 */
    char        *title_fontname;            /* タイトルフォント名 */
    char        *title_fontbold;            /* タイトルフォント太さ */
    char        *title_fontitalic;          /* タイトルフォント傾斜 */
    int         title_fontsize;             /* タイトルフォントサイズ */
    XFontStruct *title_font;                /* タイトルフォント構造体 */
    char        *title_color;               /* タイトルフォント色 */
    unsigned long title_pixel;              /* タイトルフォントピクセル値 */
    GC          title_gc;                   /* タイトルフォントGC */
    char        *label_fontname;            /* ラベルフォント名 */
    char        *label_fontbold;            /* ラベルフォント太さ */
    char        *label_fontitalic;          /* ラベルフォント傾斜 */
    int         label_fontsize;             /* ラベルフォントサイズ */
    XFontStruct *label_font;                /* ラベルフォント構造体 */
    char        *label_color;               /* ラベルフォント色 */
    unsigned long label_pixel;              /* ラベルフォントピクセル値 */
    GC          label_gc;                   /* ラベルフォントGC */
    char        *tick_fontname;             /* 目盛りフォント名 */
    char        *tick_fontbold;             /* 目盛りフォント太さ */
    char        *tick_fontitalic;           /* 目盛りフォント傾斜 */
    int         tick_fontsize;              /* 目盛りフォントサイズ */
    XFontStruct *tick_font;                 /* 目盛りフォント構造体 */
    char        *tick_color;                /* 目盛りフォント色 */
    unsigned long tick_pixel;               /* 目盛りフォントピクセル値 */
    GC          tick_gc;                    /* 目盛りフォントGC */
    char        *ticksub_fontname;          /* 目盛り(上付)フォント名 */
    char        *ticksub_fontbold;          /* 目盛り(上付)フォント太さ */
    char        *ticksub_fontitalic;        /* 目盛り(上付)フォント傾斜 */
    int         ticksub_fontsize;           /* 目盛り(上付)フォントサイズ */
    XFontStruct *ticksub_font;              /* 目盛り(上付)フォント構造体 */
    char        *ticksub_color;             /* 目盛り(上付)フォント色 */
    unsigned long ticksub_pixel;            /* 目盛り(上付)フォントピクセル値 */
    GC          ticksub_gc;                 /* 目盛り(上付)フォントGC */
    int         draw_mode;                  /* 描画モード */
    int         draw_size;                  /* 描画サイズ */
    float       *draw_buffer;               /* 描画バッファ */
    int         draw_ptr;                   /* 描画ポインタ */
    int         hold;                       /* 設定値の保持フラグ */
    int         max_exp;                    /* 目盛りのスケール値 */
    int         graph_type;                 /* グラフの型 */
    /* private */
    int         Ww;                         /* ウィンドウ幅 */
    int         Wh;                         /* ウィンドウ高 */
    int         Ax;                         /* 描画領域x座標 */
    int         Ay;                         /* 描画領域y座標 */
    int         Aw;                         /* 描画領域幅 */
    int         Ah;                         /* 描画領域高 */
    int         Rx;                         /* グラフ領域x座標 */
    int         Ry;                         /* グラフ領域y座標 */
    int         Rw;                         /* グラフ領域幅 */
    int         Rh;                         /* グラフ領域高 */
    int         Ox;                         /* プロット領域x座標 */
    int         Oy;                         /* プロット領域y座標 */
    int         Ow;                         /* プロット領域幅 */
    int         Oh;                         /* プロット領域高 */
    GC          decogc;
    XFontStruct *font;
} GrAttr;

#define GR_MODE_NORMAL                  1
#define GR_MODE_ADD                     2
#define GR_MODE_FLOW                    3

#define GR_FONT_FORMAT "-*-%s-%s-%c-normal--*-%d-*-*-*-*-*-*"
#define GR_FONT_NAME_RANGE { "helvetica", "times", "lucida", "courier", "symbol", NUL
 L}

#define GR_DEFAULT_FONTNAME             "helvetica"
#define GR_DEFAULT_FONTBOLD             "medium"
#define GR_DEFAULT_FONTITALIC           "roman"
#define GR_DEFAULT_FONTSIZE             10
#define GR_DEFAULT_LABEL_FONTNAME       GR_DEFAULT_FONTNAME
#define GR_DEFAULT_LABEL_FONTBOLD       GR_DEFAULT_FONTBOLD
#define GR_DEFAULT_LABEL_FONTITALIC     GR_DEFAULT_FONTITALIC
#define GR_DEFAULT_LABEL_FONTSIZE       10
#define GR_DEFAULT_TITLE_FONTNAME       GR_DEFAULT_FONTNAME
#define GR_DEFAULT_TITLE_FONTBOLD       "bold"
#define GR_DEFAULT_TITLE_FONTITALIC     GR_DEFAULT_FONTITALIC
#define GR_DEFAULT_TITLE_FONTSIZE       14
#define GR_DEFAULT_TICK_FONTNAME        GR_DEFAULT_FONTNAME
#define GR_DEFAULT_TICK_FONTBOLD        "medium"
#define GR_DEFAULT_TICK_FONTITALIC      "roman"
#define GR_DEFAULT_TICK_FONTSIZE        10
#define GR_DEFAULT_TICKSUB_FONTNAME     GR_DEFAULT_FONTNAME
#define GR_DEFAULT_TICKSUB_FONTBOLD     "medium"
#define GR_DEFAULT_TICKSUB_FONTITALIC   "roman"
#define GR_DEFAULT_TICKSUB_FONTSIZE     8
#define GR_DEFAULT_DRAW_MODE            GR_MODE_NORMAL
#define GR_DEFAULT_MAX_EXP              4
#define GR_DEFAULT_LINE_STYLE           "-"
#define GR_DEFAULT_LINE_WIDTH           0

#define GR_DEFAULT_GRAPH_TYPE           GR_TYPE_XY

#define GR_DASH_PATTERN_DOT_LEN         2
static char gr_dash_pattern_dot[] = {3, 3};
#define GR_DASH_PATTERN_DOTDASH_LEN     4
static char gr_dash_pattern_dotdash[] = {9, 3, 3, 3};

GrAttr  gGraph[GR_MAX_GRAPH];           /* グラフ情報 */
int gPSOnlyMode;                        /* PS出力モード */

#define GrGetWindow(gr) (gGraph[(gr)].win)
#define GrSetWindowSize(gr, w, h) (gGraph[(gr)].Ww = (w), gGraph[(gr)].Wh = (h))
#define GrSetPSOnlyMode() (gPSOnlyMode = True)
#define GrResetPSOnlyMode() (gPSOnlyMode = False)

/*
 * プライベート定数の定義
 */
/* 目盛り定数定義 */
#define GR_TICK_NUM_TICKS               5       /* Major目盛りの数 */
#define GR_TICK_MAX_TICKS               20      /* Major目盛りの最大数 */
#define GR_TICK_MAX_TICKS_MINOR         5       /* Minor目盛りの最大数 */
#define GR_TICK_MAX_ALL_TICKS           ((GR_TICK_MAX_TICKS)*(GR_TICK_MAX_TICKS_MINOR
```

```
 )) /* 目盛りの最大数 */
#define GR_TICK_LENGTH                  0.01    /* Major目盛りの相対長さ */
#define GR_TICK_MINOR_LENGTH            0.005   /* Minor目盛りの相対長さ */

#define GR_POS_DEFAULT_X                0.01    /* ウィンドウの左上の相対位置 */
#define GR_POS_DEFAULT_Y                0.01    /* ウィンドウの左上の相対位置 */
#define GR_POS_DEFAULT_WIDTH            0.98    /* ウィンドウの左上の相対幅 */
#define GR_POS_DEFAULT_HEIGHT           0.98    /* ウィンドウの左上の相対高さ */
#define GR_POS_LABEL_HEIGHT             19      /* ラベル領域の高さ(pixel) */
#define GR_POS_XTICKS_HEIGHT            20      /* x目盛りの高さ(pixel) */
#define GR_POS_YTICKS_WIDTH             28      /* y目盛りの幅(pixel) */
#define GR_POS_XTICKS_MARGIN            0.01    /* x目盛りの相対マージン */
#define GR_POS_YTICKS_MARGIN            8       /* x目盛りのマージン */
#define GR_POS_TITLE_HEIGHT             5       /* タイトル領域の高さ */
#define GR_POS_GRAPH_AREA_LEFT_MARGIN   (GR_POS_LABEL_HEIGHT) /*  */
#define GR_POS_GRAPH_AREA_RIGHT_MARGIN  5       /*  */
#define GR_POS_GRAPH_AREA_TOP_MARGIN    8       /*(GR_POS_LABEL_HEIGHT)*/
#define GR_POS_GRAPH_AREA_BOTTOM_MARGIN (GR_POS_LABEL_HEIGHT) /*  */
#define GR_POS_PLOT_AREA_LEFT_MARGIN    (GR_POS_YTICKS_WIDTH) /*  */
#define GR_POS_PLOT_AREA_RIGHT_MARGIN   5       /*  */
#define GR_POS_PLOT_AREA_TOP_MARGIN     10      /*  */
#define GR_POS_PLOT_AREA_BOTTOM_MARGIN  (GR_POS_XTICKS_HEIGHT) /*  */
#define GR_POS_METER_TICK_MARKS         0.90    /*  */
#define GR_POS_METER_WIDTH_MARGIN       0.10    /*  */
#define GR_POS_METER_HEIGHT_MARGIN      0.05    /*  */
#define GR_POS_METER_MAJOR_TICKS1       0.80    /*  */
#define GR_POS_METER_MAJOR_TICKS2       0.70    /*  */
#define GR_POS_METER_MINOR_TICKS1       0.78    /*  */
#define GR_POS_METER_MINOR_TICKS2       0.72    /*  */
#define GR_POS_METER_STING1             0.65    /*  */
#define GR_POS_METER_STING2             0.00    /*  */
#define GR_COLOR_LOW                    (-0.25) /*  */
#define GR_COLOR_HIGH                   (0.15)  /*  */

#define Free(a)                 ((a) ? ((void)free(a),a=NULL): NULL)

#ifndef exp10
#define exp10(a)  pow(10, (a))
#endif

/*
 * 関数の宣言
 */

#endif /* end of __graph_h__ */
```