ELSEVIER

Global and Planetary Change 13 (1996) 217–223

GLOBAL AND PLANETARY CHANGE

A standard format for reporting PILPS experiments

Jan Polcher [a,*], Yaping Shao [b]

[a] *Laboratoire de Météorologie Dynamique du CNRS, Paris, France*
[b] *Climatic Impacts Centre, Macquarie University, Sydney, Australia*

Received 5 May 1995; accepted 28 August 1995

Abstract

In model inter-comparisons one major obstacle is the format of the reported data. To facilitate the analysis of the results from the different models they should all be reported in the same format. This requires that the format is flexible enough to be easily implemented into the different land surface schemes. It should also be fault tolerant and allow a few consistence checks to avoid erroneous data to be submitted. The present note describes such a format developed for the PILPS project. After discussing the aim of this new format we present the routines which are used to write and read data. Finally two applications are described which allow the user to perform a number of consistency checks on the data.

1. Introduction

Until now results of PILPS experiments were reported in standard FORTRAN formats which varied from one experiment to another. This meant that each time the exact format had to be given and the variables needed to be reported in a specified order and for any variable that was not existent in a scheme zeros had to be filled in. This method is simple but not flexible nor fault tolerant. PILPS central had to cope with wrong file formats and errors partly due to the frequent changes in output requests.

A new format for reporting PILPS output was developped for the RICE and PILPS workshop. It's aim is to reduce the problems described above while keeping the reporting procedure simple and portable over a wide range of computers (So far it has been ported to SUN, SGI, HP, IBM and CRAY computers). This method will be able to handle files containing different numbers of variables and over various temporal sequences, it will also be able cope with a random order for the variables. In order to achieve this, variables will be preceded by their name and unit and the file will contain a header giving informations on the temporal structure of the data. The resulting file is still in ASCII and can thus be send by mail or edited. The computer efficiency of the format has not been an issue in its design.

It was necessary in the new format to define standard variable names and units. We have chosen new names and new units which are more rational than those previously defined. The correspondence of names in the old and new formats is given in section 6.

This new format will certainly evolve but from now on it will be backward compatible: the software should be able to read files from version 1.0 of the PILPS format at any time. There is a range of data formats used in the climate community that fulfill the requirements of the PILPS experiments (NCSA, 1989; Rew and Davis, 1990; Drach and Mobley, 1994). We did not use one of those because they are

* Corresponding author.

linked to the UNIX operating system and it would be much too complex to port them to the systems used by the various groups participating in PILPS. Beside these formats are quiet complex as they are designed to handle the huge amounts of data produced by general circulation models.

We have written a set of subroutines and functions that write and read the PILPS files. They are in standard FORTRAN and should compile on any system. You may either implement the output routines in your land surface code or write a program that transforms the data into the PILPS format. An example is provided for transforming the format used in the Cabauw experiments into this new format (tonewfmt.f).

Going with this format is a program (checkpilps.f) that checks the consistency of the reported variables along the criteria proposed by Chris Milly (pers. comm., 1994). Thus once you have prepared your PILPS file you can make sure that it does not contain any inconsistency. **It will not validate or invalidate your results it is only meant to help verifying your data.** This program is also an example of how one uses the reading subroutines going with this format.

This report gives a brief description of the subroutines and functions written so far for the PILPS format. All these routines have been tested but it cannot be excluded that they may still contain errors. Please report any problems encountered to the authors. There is a lot of room for improvements in the present version of the software, some points are described below. If you should write new routines or improve the existing ones please report them so others can share the progress.

2. Writing files

We present here the different routines contained in the file pilpsfmt.f and which are used for writing a PILPS format file. The concept is simple; one subroutine opens the file and writes the header, then one by one the variables over the reporting period are appended with their name and unit, and finally the file is closed.

For those who in their land surface scheme write the variables to the files as they are produced during the integration, a little change is required before implementing the PILPS format. You will have to store over the integration the values in memory and write all in one block once all the data for the file has been produced. This change is to your advantage as your scheme will run faster.

The pilpsfmt.f contains also an error handling routine which is used throughout the code.

In the following subsections variables which are an input to a routine will be designated by a **I** while those returned will be designated by **O**.

2.1. Pilpsint

```
SUBROUTINE pilpsint (fname, nitnb,
tayear, taday, tatime, imestp, hdrl,
hdr2, hdr3)
```

The subroutine pilpsint opens a PILPS standard output file on the FORTRAN unit given by unitnb. This variable will be used throughout the output routines to recognize the file and distinguish it from other PILPS files that might be opened. This routine also writes the start year, start day, start time and integration time step for the records, as well as 3 headers. These headers contain the schemes' name, a description of the experiment and the name and e-mail address of the participant. No special format has yet been defined for these headers but the information contained has to be redundant with file name as defined in the PILPS instructions.

fname:	(I)	Output file name (CHARACTER * 80)
unitnb:	(I)	FORTRAN unit number of file created (INTEGER)
stayear:	(I)	Year reported (INTEGER)
staday:	(I)	Julian day of the current year at which this report starts (values are between 1 and 365) (INTEGER)
statime:	(I)	Time of the day at which this report starts in a four integer format (EX:18 hours = 1800 or 18:30 = 1830 we suppose that it is at 00 seconds)(INTEGER)
timestp:	(I)	Time in seconds between values (INTEGER)
hdr1:	(I)	Header 1; scheme name (CHARACTER * 80)
hdr2:	(I)	Header 2; description of experiment (CHARACTER * 80)
hdr3:	(I)	Header 3; scheme holder name, e-mail address (CHARACTER * 80)

For monthly data we define the timestep as the average lenght, in seconds, of a month: $60 \times 60 \times 24 \times 365/12 = 2628000$.

2.2. Pilpswrt

`SUBROUTINE pilpswrt (unitnb, varname, varunit, var, length)`

The subroutine pilpswrt write a PILPS variable var to the PILPS output file opened by pilpsint and designated by the value of unitnb. This routine will check if the name of the variable specified in varname exists (See section 6 for the list of known variables) and that the units given by varunit match.

unitnb:	(I)	FORTRAN unit number of file created by pilpsint (INTEGER)
varname:	(I)	Standard name of the variable (CHARACTER* 8)
varunit:	(I)	Standard unit for varname (CHARACTER* 8)
var:	(I)	Array containing the values of this variable (REAL)
length:	(I)	Length of the array (INTEGER)

2.3. Pilpscl(O)

`SUBROUTINE pilpsclo (unitnb)`

The subroutine pilpsclo closes the file opened by pilpsint.

unitnb:	(I)	FORTRAN unit number (INTEGER)

2.4. Pilpserr

`SUBROUTINE pilpserr (modname, errtype, message)`

The subroutine pilpserr handles the error messages that are recognized in the PILPS format. Three types of errors can be distinguished: Minor errors which are just reminders to the user, warnings which are serious problem but that the subroutine can work around and finally the fatal errors that lead to the crash of the program.

modname:	(I)	Name of the calling module (CHARACTER* 8)
errtype:	(I)	Type of error (minor,warning,fatal) (CHARACTER* 8)
message:	(I)	Message to be printed (CHARACTER* 80)

2.5. Xblk

`INTEGER FUNCTION xblk(str)`

The function xblk returns the length of a character string. Such a function is included on most systems (lnblnk usually) but we prefer to have our own version as the package might be used on computers that do not have it.

str:	Character string of arbitrary length

3. Reading files

This package is included in the file pilpsrea.f and is not only for reading the file but also for retrieving all kinds of information on the contained time series. The algorithm is rather simple and relies on the fact that the files are small enough so that they can be loaded into the main memory of the computer. This chunk of memory is defined in the common block declared in the file pilpsrea.h. This software can handle more than one file, you may thus work on a few (depending on the size of the memory of your computer) files at the same time without having to read them more than once.

You will find one subroutine (pilpsload) which actually reads the file and places it in the common block. Then there is a series of function that look through the common for finding the piece of information you need. The function presented here do not yet allow to extract all the information included in the PILPS files, a few remain to be written.

All functions in this package return 0 if they have been successful in retrieving the information or else 1.

3.1. Pilpsloa

`SUBROUTINE pilpsloa (fname, unitnb, fid)`

This subroutine will load the named file into the common block defined in file pilpsrea.h and give it an identification number (fid). This identification will be used later to access the data in memory. It is different from the FORTRAN unit number specified in unitnb because fid is determined by the software whereas the unit might have to be chosen in function of other constraints in the FORTRAN code (other I/O routines). Afterwards the content of all read files can be accessed with the functions described below.

fname:	(I)	Input file name (CHARACTER* 80)
unitn:	(I)	FORTRAN unit number of file read (INTEGER)
fid:	(O)	File identification used to refer to that file (INTEGER)

3.2. Pilpsget

`INTEGER FUNCTION pilpsget (fid, varname, varunit, var, length)`

This function allows the user to retrieve the variable named **varname**. It's unit will be put in the variable **varunit** and the values in the array **var** which will be of size **length**. **fid** identifies the section of the buffer that stores the files which is to be accessed. This function will return 0 if everything went fine and 1 else.

fid:	(I)	Identification of the file (INTEGER)
varname:	(I)	Name of the variable this function is going to look for (CHARACTER* 8)
varunit:	(O)	The units of the variable will be returned (CHARACTER* 8)
var:	(O)	Array in which the data will be returned (REAL)
length:	(O)	Length of array var (INTEGER)

3.3. Pilpstim

`INTEGER FUNCTION pilpstim (fid, stayear, staday, statime, timestp, length)`

This function allows the user to retrieve the time information about the time series.

fid:	(I)	Identification of the file (INTEGER)
stayear:	(O)	Starting year of report (INTEGER)
staday:	(O)	Starting day (INTEGER)
statime:	(O)	Starting time (INTEGER)
timestp:	(O)	Time step (INTEGER)
length:	(O)	Length of time series (INTEGER)

3.4. Pilpslis

`INTEGER FUNCTION pilpslis (fid, varname, varunit, nbvar)`

This function returns the list of all variables and their units contained in the file named **fid**.

fid:	(I)	Identification of the file (INTEGER)
varname:	(O)	Array of names of variables (CHARACTER* 8)
varunit:	(O)	Array of units for the variables (CHARACTER* 8)
nbvar:	(O)	Number of variables in file (INTEGER)

3.5. Pilpshdr

`INTEGER FUNCTION pilpshdr (fid, file, desc, team)`

This function returns the headers informations of the file named fid.

fid:	(I)	Identification of the file (INTEGER)
file:	(O)	Original file name (CHARACTER* 80)
desc:	(O)	Description of run (CHARACTER* 80)
team:	(O)	Team having submitted this file (CHARACTER* 80)

4. Some applications of the PILPS format

In this section we present two programs that use all of the subroutines that have been presented above.

They can be considered as examples for the application of the PILPS format but they are also useful.

The first program allows you to transform the file containing the daily mean values of the Cabauw experiment into a PILPS file. The second program will check the consistency of the data you have put into your file and will produce a file containing the monthly means of the data. This last one has to be applied to your data **before** you send it to PILPS central.

4.1. *Tonewfmt*

This program transforms the daily values written in the format specified for the Cabauw experiments into the new PILPS format. It will ask you for two new pieces of information which were not included in the old format:

- A description of the run. For the time being no special format is defined but some PILPS experiments might ask you to put specific information in this item. (e.g.: Cabauw control exp.) (CHARACTER * 80)
- The number of the year you report. Is the data you report from the second or third year of your integration? (e.g.: 3) (INTEGER)

This program does not apply to the file of monthly mean or time step values, it will lead to an error if the daily values are not used.

4.2. *Checkpilps*

This program applies some of the tests proposed by Chris Milly (pers. comm., 1994) to the file of daily mean values you have produced with the PILPS format. It also computes the monthly mean values for all the variables present in the original file so you do not need to compute them in your model. The time-step is set to 2628000s. As not all tests are meaningful for a time series shorter than a year; this program will not perform all tests if it finds less than 365 values in the file.

The following tests are carried out:

- Are the sign conventions for the different fluxes respected?
- If possible, it will compute the mean value for the latent heat of evaporation and check if it is within the range [0.95L, 1.02L] ([minfac $\times$ 2.5 10^6, minfac $\times$ 2.5 10^6]).
- Is precipitation larger or equal to evaporation?
- Is water conserved? (within 3 mm/yr = waterbal)
- Is energy conserved? (within 3 W/m^2 = enerbal)

Please apply this program to all the files you produce with the PILPS format. If it reports some problems it means that there is an inconsistency between your data and the tests performed and not that there is an error in your model. Nevertheless we would like you to investigate its origin as it may be caused by a special feature of your code which we would like to know about. Do not forget that a problem reported by checkpilps does not mean that your scheme is wrong!

These tests are at present very crude but they can be refined or completed. The bounds chosen for energy and water balance are arbitrary and their values will decrease with time as schemes improve. checkpilps is a program that will evolve as the community defines new criteria.

4.3. *Makefile*

This file is for people working on UNIX systems. It will allow to compile the libraries and programs with the make command. You may have to change the compiler used depending on your system.

For non UNIX users this file tells you how the different files have to be assembled.

5. Things left to do

Because this is only the first version of the PILPS format, it may still contains a few bugs which need to be corrected and there is a scope for improvement in all routines. There are still a few other functionalities which are needed but for which the code has not yet been written. The main points are:

- Introduce a compression system in the format that is compatible with the requirement that the file remains ASCII,
- Improve the performance of the writing algorithm,
- Complete the list of variables known by the PILPS format and given in section 6,
- Allow pilpswrt to recognize other units and carry out the conversion,
- Write a function for getting the exact date (year,

month, day and time) of each point in the time series once it is read,

- Extend tonewfmt to the formats of PILPS phase 1 and to the files containing the time step values,
- Improve the tests carried out in the checkpilps program and complete them with a few others,
- Write interfaces between the PILPS format and the main graphic packages used in the PILPS community. This is an important point as it will allow the modelers to have a quick look at their data before sending it off to PILPS central.

6. List of variables

6.1. Energy fluxes

1	SolDn	W/m^2	downward solar radiation
2	SolAbs	W/m^2	absorbed solar radiation
3	LWDn	W/m^2	incoming long wave radiation
4	Rnet	W/m^2	net radiation
5	SH	W/m^2	sensible heat flux
6	LH	W/m^2	latent heat flux
7	LHsoil	W/m^2	latent heat flux from bare soil
8	LHcanopy	W/m^2	latent heat flux from wet vegetation surface
9	LHtrans	W/m^2	latent heat flux due to transpiration

6.2. Temperature fluxes

20	FTsoil0	W/m^2	ground heat flux
21	FTsoil1	W/m^2	heat flux from soil layer 1 to layer 2
22	FTsoil2	W/m^2	heat flux from soil layer 2 to layer 3
23	FTsoil3	W/m^2	heat flux from soil layer 3 to out layer

6.3. Water fluxes

40	Pr	$kg/m^2/d$	precipitation
41	Evap	$kg/m^2/d$	Evaporation
42	Esoil	$kg/m^2/d$	Bare soil evaporation
43	Ecanopy	$kg/m^2/d$	interception loss
44	Etrans	$kg/m^2/d$	transpiration
45	FWsoil0	$kg/m^2/d$	infiltration rate, (vertical water flux)
46	FWsoil1	$kg/m^2/d$	vertical water flux from soil layer 1 to layer 2
47	FWsoil2	$kg/m^2/d$	vertical water flux from soil layer 2 to layer 3
48	FWsoil3	$kg/m^2/d$	vertical water flux from soil layer 3 to outlayer
49	Drain	$kg/m^2/d$	drainage, vertical water flux out bottom of root zone
50	ROsoil0	$kg/m^2/d$	surface (overland-flow) water runoff
51	ROsoil1	$kg/m^2/d$	runoff (lateral flow) from soil layer 1
52	ROsoil2	$kg/m^2/d$	runoff (lateral flow) from soil layer 2
53	ROsoil3	$kg/m^2/d$	runoff (lateral flow) from soil layer 3
54	Snowm	$kg/m^2/d$	snow melting
55	ROsoilr	$kg/m^2/d$	runoff (lateral flow) from root zone

6.4. Temperature

60	Tair	K	air temperature at reference level
61	Trad	K	surface radiation temperature
62	Tcanopy	K	canopy surface temperature
63	Tsoil0	K	bare soil surface temperature
64	Tsoil1	K	soil temperature at a depth of 0.05m
65	Tsoil2	K	soil temperature at a depth of 0.5m
66	Tsoil3	K	soil temperature at a depth of 1m
67	Tsoilr	K	soil temperature in root zone

6.5. Moisture

80	Mcanopy	kg/m^2	interception storage on canopy
81	Mroot	kg/m^2	Root-zone soil water
82	Msoil1	kg/m^2	Soil moisture in the top 0.1 m
83	Msoil2	kg/m^2	Soil moisture from 0.1 to 1 m
84	Msoil3	kg/m^2	Soil moisture from 1 to 1.5 m
85	Wsoil1	m^3/m^3	volumetric soil water content in the top 0.1 m
86	Wsoil2	m^3/m^3	volumetric soil water content from 0.1 to 1 m
87	Wsoil3	m^3/m^3	volumetric soil water content from 1 to 1.5 m
88	Snowa	kg/m^2	snow amount

6.6. Others

100	Uair	m/s	u-component of wind speed
101	Vair	m/s	v-component of wind speed
102	Albedo		albedo
103	Qair	kg/kg	specific humidity of air at reference level
104	Pres	N/m^2	atmospheric surface pressure
105	Rair	s/m	aerodynamic resistance
106	Rstomat	s/m	stomatal resistance
107	Rsoil	s/m	soil resistance
108	alphaTr		alpha parameter for transpiration
109	betaTr		beta parameter for transpiration
110	alphaBs		alpha parameter for bare soil evaporation
111	betaBs		beta parameter for bare soil evaporation

6.7. Correspondence between variable names in the old and new formats

In the new format there is no distinction between the names and the units of the variables for monthly, daily and time step values. We report here only the correspondence of names for daily values as in this case units in the old format match those of the new format.

old formats	new format
APPT	Pr
AE	Evap
AR_d	Drain
AR_sAR_i	ROsoil0 + ROsoilr
AXM	Snowm
AS_i	Mcanopy
AS_r	Mroot
ASWET	Msoil1
AS_f	Snowa
ATRAD	Trad
ATC	Tcanopy
ATU	Tsoil1
ATL	Tsoil2
AABS	SolAbs
AXNET	Rnet
AE_t	LH
ASENS	SH
AALBEDO	Albedo

References

Drach and Mobley, R., 1994. DRS users's guide. Univ. Calif., LLNL, Livermore, Tech. Rep., 16, PCMDI.

NCSA, 1989. NCSA HDF calling interfaces and utilities, version 3.0. Natl. Center Supercomput. Applic., Univ. Illinois, Urban-Champaign.

Rew, R. and Davis, G., 1990. NetCDF: an interface for scientific data access. IEEE Comput. Graph. Applic., 10(4): 76–82.