Local Variable Storage

1. Identification of the Proposed Change

1.1 Title: Local Variable Storage

1.2 MDC Proposer and Sponsor:

Proposer:
X11/SC13/TG15: Local Variable Storage

Sponsor:
X11/SC13/TG15: Local Variable Storage

Editor: Frederick D. S. Marshall
1660 South Columbian Way (ISC)
Seattle, WA 98108-1597
O: (206) 764-2283 H: (425) 430-0932
fax: (206) 764-2923
toad@isc-sf.va.gov FORUM

1.3 Motion:

The proposer recommends that this document be accepted by MDC as a Type A extension.

1.4 History:

March 1998	X11/SC13/TG15/97-3	<Current Document> Supersedes X11/SC13/TG15/95-6. Revised: updated doc #, motion, history, pros & cons. Proposed as MDC Type A.
March 1996	X11/SC13/TG15/95-6	Local Variable Storage. Supersedes X11/SC13/TG15/95-2. Revised: includes approved amendments. Accepted as SC13 Type A: 15:0:5.
4 June 1995	X11/SC13/TG15/95-2	Local Variable Storage. Supersedes X11/SC13/TG15/95-1 and X11/SC15/94-23. Revised: change 7.1.1 to reflect local storage on disk. Accepted as SC13 Type B with amendments: 16:3:3. Amendments: make conformance specify min, not max; add explicit portability statement.
January 1995	X11/SC13/TG15/95-1	Local Variable Storage. Supersedes X11/SC15/TG15/95-1. New document number. Accepted as SC13 Type C: 21:0:2.
	X11/SC15/TG15/95-1	Local Variable Storage. New approach: no portability limit. Proposed as Type C. TG15 transferred to SC13: without dissent.
May 1994	X11/SC15/94-23	Process Specific Globals V2. Supersedes X11/SC1/90-77. Improved formalism. Remanded to new TG15 by SC15: 22:5:0. SC15/TG15 formed to deal with document.
October 1990	X11/SC1/90-77	Original Proposal. Created new variable entities, with new syntax.

1.5 Dependencies

None.

2. Justification of Proposed Change

2.1 Needs

We have a problem. Although on some level we are aware that caching and virtual memory have changed the actual distinctions between locals and globals, we still tend to think of locals as residing in RAM and globals on disk. This is a myth. We are mentally clinging to the past.

This problem results in confusion about the true differences of locals from globals. Historic enhancements in M implementations have made the issue of residing in RAM versus on disk more a question of frequency of access than of local versus global. To simplify a complex subject, the most used globals and locals will be in RAM--in cache or in the RAM portion of the symbol table--while the remainder will be on disk--in the global or virtual memory portions, respectively. To believe otherwise is to make coding decisions based on false premises.

The standard perpetuates this myth in the storage space restrictions chapter of the M Portability Requirements section. There is no standard minimum for global storage space, but there is for local variable storage. The assumption here is that global space resides in a vast pool, effectively limitless, while local space is a precious resource that must be doled out and protected: that globals are on disk, locals in RAM.

We submit that this distinction is inappropriate. If local variable storage restriction is designed to protect precious RAM resources, then it applies to both globals and locals or to neither, but not just to one.

This myth results in practical problems. The periodic efforts to repair or replace $STORAGE burn up MDC resources. The artificial restriction on local variable storage results in such proposals as Process-Specific Globals, as developers struggle to liberate the true characteristics of locals--privacy and transience--from limits forever doomed to be confining. Developers waste precious development time deciding between the use of locals and temporary globals on the basis of their best guesses as to how much symbol table space will be left at various points in their routines. If we retain limits on locals, then we shall always tie up resources raising the limit by new "reasonable" increments in an attempt to alleviate cramped conditions that are in truth wholly unnecessary.

The obvious solution to this problem is to eliminate explicit limits on local variable storage. Vendors will divide the symbol table between RAM and disk however best optimizes performance. M programmers can stop worrying about it, since the standard will then acknowledge the truth that they don't really have control over it now anyway.

The true distinctions between locals and globals will more clearly emerge in the standard. Locals are private and transient. Globals are shared and permanent. The mental shift this change will cause is needed, welcome, and overdue.

2.2 Existing Practice in Area of the Proposed Change

Many implementations already use caching and virtual memory, and cap the local variable storage at some level that exceeds the standard limit. Developers currently struggle to remain within the symbol table limits, and use temporary overflow globals to handle transient, private data that will not fit; some developers end up developing scheduled background jobs that exist solely to clean up these "temporary" globals when the normal cleanup procedures fail.

3. Description of Proposed Change

3.1 General Description of the Proposed Change

This proposal removes the restriction on local variable storage.

3.2 Annotated Examples of Use

As a change to the environment rather than M syntax, it is difficult to show an annotated example of use. Indeed, this change is more noteworthy for what need not be coded.

3.3 Formalization

Make the following changes to MDC X11.1-1995, Section 1, Chapter 7.1.1.

Change paragraph 2 to read:

> An M routine, or set of routines, runs in the context of an operating system process. During its execution, the routine will create and modify variables that are restricted to its process. These process-specific variables may be stored in primary memory or on secondary peripheral devices such as disks. It can also access (or create) variables that can be shared with other processes. These shared variables will normally be stored on secondary peripheral devices such as disks. At the termination of the process, the process-specific variables cease to exist. The variables created for long term (shared) use remain on auxiliary storage devices where they may be accessed by subsequent processes.

Make the following changes to MDC X11.1-1995, Section 1, Chapter 3.1.

Insert a new line into the indented paragraph of part c, so that the indented paragraph will now read:

> "*xxx* version *v* conforms to X11.1-*yyyy* with the following exceptions:
> ...
> Supported Character Set Profiles are ...
> Uniqueness of the values of $SYSTEM is guaranteed by ...
> The minimum amount of local variable storage for a job is guaranteed to be ...:"

Make the following changes to MDC X11.1-1995, Section 2, Chapter 8.

Delete every sentence except the first three.

Add a new second paragraph:

> Note: in comparison to previous versions of the standard, there is no specification of local variable storage. Like global variable storage, local variable storage can be arbitrarily large. The implementation's conformance statement must specify the minimum guaranteed to be available.

4. Implementation Effects

4.1 Effect on Existing User Practices and Investments

None. This will create no problems in code that already runs correctly. If an application is counting on an error condition arising if it exceeds the current 10,000 character limit, its behavior would change: the error would no longer occur. However, this kind of consideration has not restricted previous proposals to increase the local variable storage. We anticipate no change in current user practices and investments.

4.2 Effect on Existing Vendor Practices and Investments

None. We have confirmed that MSM, VAX DSM, GT.M, PFCS, and MGlobal already make use of virtual memory, and that DTM does not.

4.3 Techniques and Costs for Compliance Verification

The presence of the conformance clause pertaining to available local variable storage will itself equate to compliance with the standard.

4.4 Legal Considerations

None expected.

5. Closely Related Standards Activities

5.1 Other X11 Proposals Under Consideration

None.

5.2 Other Related Standards Efforts

None.

5.3 Recommendations for Coordinating Liaison

Unnecessary.

6. Associated Documents

None.

7. Issues, Pros and Cons, and Discussion

The history of the original document is unclear, except that it was proposed by Greg Kreis, and subsequently languished. Ben Bishop later created a new document to more formally propose the approach Greg Kreis intended.

June 1994

In SC15, Mr. Bishop moved that X11/SC15/94-23 (Process Specific Globals V2) be accepted as Subcommittee type B: Rick Marshall seconded.

Pro

1. Removes storage limitation that locals are subject to [15].
2. Alleviates clutter on system [15].
3. Prevents other jobs from killing your global [9].
4. TP effects are clear [3].
5. Desired functionality [6].

Con

1. Adds clutter to language [17].
2. Unnecessary [5].
3. TP effect is unclear [13].
4. Wrong approach [14].
5. Makes error investigation more difficult [8].
6. Cannot be passed by reference [11].
7. Destroys conceptual integrity [13].
8. TP effects violate principle of least astonishment [13].
9. Should be OO-related [1].

Larry Ruh moved to remand to task group; Carl Bauer seconded. A straw poll voted 14-8 in favor of this functionality. The motion to remand passed 22-5 by show of hands. TG15 was created; Mr. Marshall was appointed chair. A written ballot was taken to provide guidance to the task group.

SC15/TG15 met in the hall for five minutes after the SC15 meeting. The Process

Specific Globals approach was voted on and rejected by a vote of 5 to 0 in favor of the approach taken by the current document. We believe this approach retains all factors cited as Pros while addressing eloquently all issues cited as Cons, with the possible exception of Unnecessary. However, since the straw poll endorsed the functionality, we believe this Con represents a minority view, and at any rate may have been strongly influenced by the method of achieving this functionality described in the remanded document.

January 1995

X11/SC15/TG15/95-1 was the first draft of this proposal based on the discussions of the previous meeting.

SC15/TG15 met Jan 26 and voted to advance this document in SC15..

Pro	Con
1. Removes artificial ceiling.	1. Removes guaranteed minimum.
2. Simple and elegant.	2. Insufficient conformance.
3. Does not require programmer to manage system resources.	3. Does not allow programmer to control choice of RAM vs. disk.

In SC15, before X11/SC15/TG15/95-1 could come up for a vote, Dan Bormann moved (and Chris Richardson seconded) to transfer the task group to SC13; the motion passed without dissent.

In SC13 X11/SC13/TG15/95-1 was accepted as type C 21:0:2 with the same pros and cons cited as by TG15.

June 1995

SC13/TG15 decided the document should somehow be strengthened to indicate the increased similarity between globals and locals. The document editor identified chapter 7.1.1 as the only place where the Standard ties globals to auxiliary storage. This version of the document similarly states that locals may be in primary memory or on disk, and that those on disk will be removed when the process terminates. Those who raised this issue feel this change addresses their concerns. The group voted 5-0 to propose that SC15 accept this document, X11/SC13/TG15/95-2 as Type B.

Subsequent to that meeting, Jon Diamond, who could not attend the TG15 meeting, recommended that the document be amended to strengthen the language further. First, the local variable storage conformance clause was strengthened by focusing on the minimum symbol table storage available, rather than the maximum. Second, we added text to the portability section to explicitly say the symbol table limit has been removed, that the result is that the symbol table can now be arbitrarily large, and that the conformance clause of each implementation will specify what the implementation provides.

In SC13, Rick Marshall moved that SC13 accept X11/SC13/TG15/95-2 as a Subcommittee Type B document, superseding X11/SC13/TG15/95-1 and X11/SC15/94-23. Rick Marshall, seconded by Rod Dorman, then moved to amend the document as described above. Roger Partridge, seconded by Rick Marshall, then moved to amend the motion to strike the clause ", and any stored on auxiliary storage will be removed" that had been added to 7.1.1 by TG15. The motion to amend carried by voice vote, and the main motion then carried 16:3:3. The pros are recorded in the minutes of SC13, but the cons are omitted. Since three cons were cited in the vote, they may be the same three cons used in the votes on the previous two versions of this document:

Pro	Con
1. Has been implemented. [3]	1. Removes guaranteed minimum. [5]
2. Elegant solution. [5]	2. Insufficient conformance. [1]
	3. Does not allow programmer to control choice of RAM vs. disk. [4]

TG17 feels con 3 is not valid, as programmers don't have as much control now as they think they do, and several of those citing this con who were questioned informally after the vote acknowledged this. Con 2 begs the question of what is missing, and no amendment to change the conformance clause has been forthcoming. Con 1 is the more troublesome of the three, but implementors who have been asked about this say the guaranteed minimum cannot actually be guaranteed on layered systems (which account for most M platforms today), since memory allocation is shared with non-M software outside the M implementor's control. The majority view has been that removing the stated minimum will still leave programmers protected by the portability note stating the intent of achieving fairly unlimited local variable storage, and by the programmers' own enthusiasm for this feature, which should help thwart any implementors seeking a narrower interpretation of the new wording.

March 1996

X11/SC13/TG15/95-6 is the amended document resulting from the SC13 vote at the June 1995 meeting. The only changes from X11/SC13/TG15/95-2 were the inclusion of the amendments added at the last vote, and updating the document number, history, and pros and cons sections (although the motion listed in the document was not properly updated).

In SC13, Rick Marshall moved that SC13 accept X11/SC13/TG15/95-6 as a Subcommittee Type A document. The motion carried 15:0:5, with the following pros and cons cited:

Pro	**Con**
1. States the obvious.	1. Can't measure local storage. [3]
2. Solves the problem for a long time.	
3. User demand. [1]	

TG15 was encouraged to see each new vote on this proposal increasingly pro. The con is a curiosity considering the M community nearly universally acknowledges that $STORAGE is almost useless, such that effectively programmers can't measure local storage now in any meaningful way.

September 1997

SC13/TG15 met and decided to produce the replacement document for X11/SC13/TG15/95-6 with revised proposal, history, and discussion sections for presentation at the March 1998 meeting. This is that document. As this will be the document editor's first MDC Type A if approved, great care has been taken on this final version to record accurately the history of this document.

8. Glossary

None

9. Appendix

None