ED 094 780 IR 000 981

AUTHOR	Gross, Jean M.
TITLE	Transportability of Courseware: Standardization of Installation Modifications.
INSTITUTION	Rutgers, The State Univ., New Brunswick, N.J. Hill Center for the Mathematical Sciences.
PUB DATE	Aug 74
NOTE	8p.; Paper presented at the Association for the Development of Computer-based Instructional Systems Summer Conference (Bellingham, Washington, August 13-15, 1974)
EDRS PRICE	MF-$0.75 HC-$1.50 PLUS POSTAGE
DESCRIPTORS	*Computer Assisted Instruction; Instructional Systems; Problem Solving; Programers; *Programing; Programing Problems; Speeches; *Standards
IDENTIFIERS	ADCIS; Assn Development Computer Based Instructional Syst; Compatability; Coursewriter

ABSTRACT

A study on the transportability of computer courseware was presented to the Association for the Development of Computer-based Instructional Systems (ADCIS) at their August 1974 meeting. This study addressed the area of minor incompatibility that can cause transportability of customized Coursewriter software at individual sites to become difficult. Nonetheless, customizing software is basically good in that it allows additional capability and greater flexibility. The disadvantages, and impact on transportability, occur when separate installations make modifications that are mutually exclusive and conflicting. A solution lies, in the forum ADCIS provides, for concerned installations to work together to coordinate customizing. This allows desired modification while minimizing the impact on courseware transportability. The author suggests that installations form a subgroup under the Coursewriter Systems Implementation Group. In general, the subgroup could examine and exchange information on modifications of this nature with an effort toward achieving some standardization or set of guidelines to minimize the possible danger. (WCM)

Transportability of Courseware: Standardization of Installation Modifications

presented at the

Association for the Development of Computer-based Instructional Systems

August 1974 Meeting

Jean M. Gross
SYSTEMS COORDINATOR
Center for Computer and Information Services
Hill Center for the Mathematical Sciences
Rutgers - the State University
New Brunswick, New Jersey 08903

One of the basic purposes of ADCIS (Association for the Development of Computer-based Instructional Systems) is fostering the exchange of computer-based instructional materials. The growth of the use of computers in the instructional process is dependent on the exchange of such materials. It is recognized that a given amount of material (critical mass) is necessary in a discipline to get CAI/CMI off the ground. This minimum amount of courseware material can be developed by each installation, but has the obvious drawbacks of duplication of effort which increases costs, thereby creating additional obstacles for budding CAI/CMI installations.

In many cases installation dependent configurations limit the transportability of most courseware (i.e. language differences, hardware incompatibilities, terminal dependencies). These incompatibilities will, in all likelihood, remain to plague CAI/CMI users for the foreseeable future. I will not address these problems in this paper, but discuss, instead, an area of minor incompatibilities, that can cause courseware transportability to become difficult. This incompatibility is in the area of the customization of Coursewriter software at individual sites. Customizing software is basically good in that it allows additional capability and greater flexibility. The disadvantages, and impact on transportability, occur when separate installations make modifications that are mutually exclusive and conflicting. A solution lies, in the forum ADCIS provides,for concerned installations to work together to coordinate customizing. This allows desired modifications while minimizing the impact on courseware transportability.

Coursewriter III course material is entered into the system by one of three methods: terminal input of source code; 'course on'; and 'auto insert'. In the first instance, the author or instructional programmer enters, line by line, all the course material. As each line is entered Coursewriter converts the op codes to an internal code for execution. At the same time, all text material is translated to 1050 line code. This translation to 1050 line code is basic to understanding how Coursewriter III processes data (i.e., courseware, student input). The 'course on' method is normally used in transferring courseware from one installation to another. In this way the internal representation (1050 line code) is copied directly to tape. The tape is forwarded and 'course on'd' at the receiving installation without any intervening translation. The last method, 'auto insert', allows source code insertion. Course material is punched in cards, put on tape and 'auto inserted'. During the process source code is translated to 1050 line code.

Regardless of which method one uses to input course material, once source code is converted to 1050 line code, courseware remains unchanged until edited by an author. When a course is executed in student mode, Coursewriter retrieves material, examines the converted op codes and determines logical action. As Coursewriter directs text to the student, it applies the appropriate translation to the 1050 line code (internal) representation. This is done by examining the terminal type of the currently signed on user. The bulk of translation is performed through the use of translate tables and one machine language instruction (TRANSLATE).

Of the three methods for inputting course material, the 'course on' process is the most feasible for exchanging courseware between installations. It would appear the 'auto insert' process could be used for transferring source code between installations. Upon further exploration one sees 'auto insert' is a one-way process. It can be used to put material on a system but it cannot be used to remove material.

In transferring courseware, it is desirable to be able to copy the current, most recently edited version from the system. 'Course off' will do this. Since 'course on/off' copies courseware in 1050 line code (internal) it is important to recognize the impact when an installation modifies its Coursewriter system such that internal 1050 code representation of control functions or characters differs from the standard representation. The modifying installation may be restricting the future transportability of their courseware. It is also possible to affect the importation of courses if the sending installations have customized their system.

Why would Coursewriter installations alter internal representations? Why is it a current problem? Presently there are a variety of terminals on the market that can be used with Coursewriter teleprocessing support. There is a significant number of (ASCII) terminals alone. The ASCII type terminals available in the marketplace run the gamut from low-cost, low-function hard copy types (e.g. Teletype ASR33) to sophisticated CRT terminals with a host of control function keys for cursor control, hard copy printer control, etc. Add to this the variety of EBCDIC and graphics terminals available, one can see the proliferation of terminals, each with new and desirable capabilities.

Several phenomena are evident with regards to the more sophisticated terminals. One, a CAI/CMI user may be sold a terminal in the full belief that all the "goody" control functions the salesperson demonstrated would be available to the Coursewriter author. Sometime after signing the sales agreement it is discovered that these special functions are unsupported. At this point, the user turns to the Coursewriter systems programmer with the request to make the functions work. Alternatively, the high-function terminals are obtained with full knowledge that Coursewriter must be modified to support special control functions. In the latter case the design of a CAI/CMI program may dictate the need for special control function capability. A request to implement terminal control functions does not, on the surface, appear to be unreasonable. There are several reasons why Coursewriter systems programmers would grant such a request: the required system modifications involve only changing the translate tables, and it pleases the user while providing expanded terminal support.

A case in point occurred at Rutgers. A user purchased Digilog CRT terminals and designed their CMI program around the use of function keys to control CRT display and hardcopy output. The user, upon finding that Coursewriter did not support these function keys, requested the necessary software modifications. The problem was investigated and modifications were made to two translate tables (the TTY to 1050 (TRTT1050) and 1050 to TTY tables (TR1050TT)).

The unmodified Coursewriter III translate tables are constructed so that special functions not supported are translated to null or fill characters. The obvious solution would be to translate desired control functions to their 1050 equivalent or some unique bit configuration. Upon investigation, it turns out changing translate tables is not quite as straightforward as one would initially expect. Coursewriter nulls out many non-supported characters before applying the output translate. To successfully implement control function or special character support, one must isolate which bit structures Coursewriter will leave unchanged so they can be used for the internal translation. It is necessary to select bit structures that are unchanged and produce a one-to-one mapping. (In the course of implementing the Digilog control functions, the author isolated three such bit structures).

How does all this affect transportability? Visualize, in the case of the Digilog control functions, what course material looks like internally. The CMI course was designed so that the CRT screen was blanked, and the cursor returned to the top left corner before presenting a new question. This is done by placing a control L (Digilog) in the first character position of each qu text. The control L (hex 30,31) is translated to 1050 line code (by the Rutgers CCIS modification) to a 3A, (see Table 1). In an unmodified system, a control L is translated to a 90 (effectively a null character). If there was no software modification or there was some way to transport source code, the presence of a control L would normally be nulled out when translated to 1050 line code. If the above courseware is 'course off'd' and 'course on'd' at another installation, it will carry with it the 3A at the beginning of each qu text. If the installation has made no translate table modifications, then the system translates the control function (3A) to a 55 (*) when making the output translation. If, though, the receiving installation has modified its system to handle special control functions, etc., and it has used any of the same internal codes, one can see the conflicts that arise. It is easy to visualize the situation where installation A has decided to make a cntrl L translate to a 3A, and installation B has decided to make a cntrl P translate to a 3A. The course from installation A transferred to installation B will trigger cntrl P everywhere it was intended to perform cntrl L. The author experienced the reverse problem when the computer center received a large course from another installation. When 'course on'd' it was discovered that where carriage returns should appear, there were tabs. This was entirely due to modifications made by the originating installation. As one can see, the greater the proliferation of installation modifications being made that affect the internal representation of code, the greater the incidence of conflict.

I believe that system modification to enhance terminal capability is a good idea. The hazard lies in each installation determining independently how to do its modification. In order to protect Coursewriter installations from setting potential pitfalls affecting exchange of courseware, I suggest that interested installations form a subgroup under the Coursewriter Systems Implementation Group. As an early effort of such a subgroup, I suggest that installations pool the current status of their modifications. Questionnaires could be sent to all Coursewriter installations requesting similar input. As an important contribution, I would like to see recommendations compiled, suggesting how translate tables should be modified, (i.e. what internal codes to use in translation). This would encourage installations to make identical and non-conflicting modifications for each desired control function or special character. In general the subgroup could examine and exchange information on modifications of this nature with an effort towards achieving some standardization or set of quidelines to minimize the possible dangers.

COURSEWRITER III V Translate Tables

ASCII to 1050 code, 1050 code to ASCII
modified by Rutgers University

```
          DC     X'[illegible]9F'
          DC     X'[illegible]A3'
          DC     X'[illegible]8E'
          DC     X'[illegible]E8'
          DC     X'[illegible]DC8E'  3.3
          DC     X'[illegible]7E28E'
          DC     X'[illegible]BFFF'
          ENTRY  TRTT1050
          DS     0F
TRTT1050  DC     X'9090902090019090909C4F4F15159090'
          DC     X'909070709039390919090902F5C109090'
          DC     X'901F6868575790907979902690089090'
          DC     X'3A3A9045903790909090333384849090'  CNTRL L CHG
          DC     X'9090648490690901990905290049090'
          DC     X'3838904390909090909032328888909090'
          DC     X'909090609081909090902A2A0D0D9090'
          DC     X'71714A4A7676909090909057908E9090'  CNTRL N CHG
          DC     X'5858626207D79090909090519000290090'  CNTRL A CHG
          DC     X'7A7A9073909590909090313113139090'
          DC     X'9090906590839090909092929080B9090'
          DC     X'90584949404090909090903490829090'
          DC     X'9090905790169090909090252570709090'
          DC     X'9090454551E1909090909090359087909090'
          DC     X'90906E6E8D8D909090909020C9090E9090'
          DC     X'F1F1904C902390909090C0C0A3A3907F'  CNTRL O CHG
          ENTRY  TR1050TT
          DS     0F
TR1050TT  DC     X'55058D554D5555CC2D5555AC555CED55'
          DC     X'1D55559C550CC5555548555555555521'
          DC     X'035555F555CA285555AA6A55EB55551B'
          DC     X'559A5A3AB3D85535555530505558155555'  CNTRL L CHG
          DC     X'3455555355D233555592725F3555550A'
          DC     X'55384855555555245555558155555FF55'
          DC     X'55658255425555C3225555A35563E255'
          DC     X'12775593555574555528905555555FF'  CNTRL N CHG
          DC     X'55058D553C5555DD5C5555A555E47D55'
          DC     X'55555514559544555548555555555555'
          DC     X'D85555FC55CA285555AA6A55EB55551B'
          DC     X'559A5A5555555578555555550558155555'
          DC     X'FA555553 55D2335555827255F355550A'
          DC     X'55534855555555845555558155555FF55'
          DC     X'55048255425555C3225555A35563E255'
          DC     X'12FF5593555534555528905555555FF'  CNTRL O CH
* THE FOLLOWING TABLES CONVERT RELATIVE CONTROL UNIT ADDRESS (0-31)
* TO USE IN XVISLIST FOR POLLING OR SELECTION.  THEY ARE ALSO USED
* TO CONVERT 3270 SCREEN POSITION (0-63) TO A START BUFFER ADDRESS
* (SBA).  CODE IS ASCII OR EBCDIC OR BOTH.

LCBEND    EQU    *
          ENTRY  LCBEND
          END
```