З

<section-header>         AN EXPERIMENTAL TIME-SHARING SYSTEM       000000000000000000000000000000000000</section-header>			335
<section-header>         Formando J. Corbato, Marjore Mcrwin-Dagget, Robert C. Daley       36         Manbridge, Massachusetts Institute of Technology       36         Commerce       36         Summerg       36         Summerg       36         Summerg       36         It is the purpose of this paper to discuss indiget mode of the slow nan-computer inter-indiget mode on of us (270) that illustrates to some or in some complexer in the orally 190° (1) that illustrates to some programs are written to the slow nan-computer uses of interest, As a result ingree on dinconster, problems below on the earby 190° (1) that illustrates to some programs are written to take avantage of larger and faster to complexers, non on targe computer uses of interest, is a sensell ingree on and nonster in the middet of a third major modifier uses have nonstare computer uses in the middet of a third major modifier uses and consulter uses been device to the programs bug usually requires several hours of computer is the improved with earby 190° (1). The some of the slow gar of an onstare computer uses in the middet of a third major modifier uses and consulter uses been device to the program bug usually requires several hours of computer is the improved with earby 190° (1). The some of an enderget of the some of an enderget of the some of an enderget of the some of the some of an enderget of the some of the so</section-header>	AN EXPERIMENTAL T	IME SHARING SYSTEM	
Fernando J. Corbató, Marjorie Merwin-Daggett, Robert C. Daley       66         Computer Center, Massachusetts Institute of Technology       67         Cambridge, Massachusetts       67         Summary       77         It is the purpose of this paper to discuss       77         parting system which has been developed to the contingues thich may be employed to makine and be analyzed for the performance limits of such a time-sharing system.       77         Introduction       The last dozen years of computer usage have a strikes, In the early 1950's, the tase language ever gravally in the construction and bean larguages were gravally in the construction and handpred computers. This process inevitably leads to more programs which are worked in the subject, in the subject in the su	AN EXCERNIENTAL TIME-SHARING SISTEM		
Computer Center, Massachusetts Institute of Technology       66         Summary       77         To is the purpose of this paper to discuss       78         briefly the need for time-sharing, some of the partine system.       77         and not one of us (RyC) that illustrates to the nearby 160's the performance and the sharing system which has been developed to the sativate of the performance and the sharing system.       78         Introduction       78       78         To last dozen yoars of computer usage have some of the activation of a time-sharing system; in the mid-180's, the advent of a computer system; in the mid-180's, the advent of a computer system; in the mid-180's, the advent of a computer system; in the mid-180's, the advent of a computer system; in the mid-180's, the advent of a computer system; in the mid-180's, the instruction by a process called the advent of a third major modification computer system; in the mid-180's, the instruction is paper, has been dwoloped the instruction is paper, has been dwoloped the instruction is paper, has been dwoloped the instruction that is paper should be taken as solely these.       78         Nuch of the time-sharing millosophy moving computer system interaction be appreciable interportation to computer system; in the mid-180's, the instruction is paper, has been dwoloped these.       78         Nuch of the the system for this paper hould be taken as solely these.       78         To inter-sharing millosophy moving computer system.       78         the interviction of y a process called the interaction more illass the interaction more interaction more interaction the second	Formando I Correto Mariaria M	Convin Deggett Babant C. Delan	
Computer Center, Massachusetts Institute of Technology Cambridge, Massachusetts 67 Cambridge, Massachusetts 67 Cam	remando J. Corbato, Marjone M	rerwin-Daggett, Robert C. Daley	
Summary Summary It is the purpose of this paper to discuss briefly the need for time-sharing, some of the scheduling algorith of one of us (72) that illustrates briefly the need for time-sharing system which has been developed for the scheduling algorith of one of us (72) that illustrates briefly the need for the performance inits of any time sharing system. This rate has been developed for the hast does near strides. In the early 1960's, the fore or performance of hardware; in the mid-1980's, the roomputer value of the advantage of larger and faster computers. Note a time-sharing philosophy, which are the side of a third major modification to computer usage: the improvement of mammark and the presently problems solely the description of the time-sharing philosophy, the present authors. Much of the time-sharing philosophy, the resent has the computer, nore completer you was been due to for debugging. Using current batch mouther of the present authors that of any finding the scheduling the scheduling the scheduling the scheduling the scheduling. Using current batch mouther of the present authors computer that has been developed in this paper, has been developed in the scheduling philosophy. The process inverticely is for the regured for debugging. Using current batch mouther of the hardware of a milding the scheduling the schedul	Computer Center, Massachu	setts Institute of Technology	
Summary It is the purpose of this paper to discuss briefly the need for time-sharing, some of the implementation problems, an experimental time contemportry IMM (700), and finally a scheduling algorithm of one of us (201) that illustrates some of the techniques which may be employed in the analyzed of the performance limits of such a time-sharing system. Introduction The last doem years of computer usage have seen great strides. In the early 1960's, the problems solved were largely in the construction the advent of compilers in win in the early 1960's, the usage languages were greatly improved with the advent of compilers in win in the early 1960's, the usage languages were greatly improved with the advent of compilers in win in the early 1960's, the usage languages were greatly improved with the advent of compilers in win in the early 1960's, the usage languages were greatly improved with the advent of compilers in win the early 1960's, the usage languages were greatly improved with the advent of compilers in win the early 1960's, the usage languages were greatly improved with the advent of a MIT computer swained the long range computer usage have study committee, chirred by J. McCarthy, inverse the studies of a third major modifi- time-sharing. Much of the time-sharing philosophy, reprise and the ong range computational the ease of the Institute, and a subsequent MIT computer study committee, chirred by J. McCarthy, inverse the studies that mass this is best to five a more process information to time- sharing, often mean ensure of this proves that the studies of the programmer definition to the starting, detern man ensure of fifterent parts of the hardware at the same line for different parts of the hardware at the same line for different parts of the hardware at the same line for different parts of the hardware at the same line for different parts of the hardware at the same function to time- sharing, often mean several persons making the hardware at the same hardw	Cambridge, N	Massachusetts	
Summary Summary It is the purpose of this paper to discuss the field of the need for time-sharing, some of the schemarging to the performance intersection are operated for the computer usage has been developed for the computer usage has been developed for the performance of hardware; in the mail-1850's, the not of computer usage has been developed for the advert of computer usage has been developed for the performance of hardware; in the mail-1850's, the nonsole, the schemarging of computer usage has been developed for the performance of hardware; in the mail-1850's, the nonsole the developed for the advert of computers; the inproved with the schemarging of the formation of antime schemarging to the advert of the schemarging of the schemarging to the advert of computers; the inproved with the advert of advertige of the schemarging of the schemarging to the schemarging of the schemarging of the schemarging to			68
Summary72It is the purpose of this paper to discuss briefly the need for time-sharing, some of the inglementation problems, an experimental time- sharing system which has been developed for the contemporty 108/7090, and finally as calculating algorithm of one of us (RJC) that illustrates some of the techniques which may be employed to that as dimentary possible with the bigger, some of the techniques which may be employed to the last decade of widespread computer use.1077ThroductionIn part, this effect has been due to the bigger, nor advanced computer, some computer section required to take advantage of larger and faster computers. the last decade of widespread computer use.1078ThroductionIn part, this effect has been due to the section required to take advantage of larger and faster computers. the usage languages were greatly improved with the advant of compluter use, the indeveloped in to chaine sharing the looptry over and has been developed in to chaine which is paper, has been developed in computer uses, the one on most large computers, to elanguaring computer use, the only to chaine which as a brief or the advantage of larger and faster computers. This process multiple cast in the programmer to attempt to debug directly at the computer use of the show mark of an MIT preliming. The presention with the work of an MIT preliming, is of the nerver end conclusions experiment parts of the hardware at the same time, the fiftiency in this paper, which sense of attempt of the fiftiency in the sense of the conclusions experiment parts of the present authors.78The due due due due taken as solely those to the intersection profess making sense of the concellusion serversent properse making to consult the same time for different taking prosent authors. <br< td=""><td></td><td></td><td>69</td></br<>			69
Summary72It is the purpose of this paper to discuss briefly the need for time-sharing, some of the tentemporary IEM 7090, and finally a scheduling tagorithm of one of us (FU) that illustrates some of the techniques which may be employed to conhance and be analyzed for the performance limits of such a time-sharing system.76Introduction78The last dozen years of computer usage have seen great strides. In the early 1950's, the usage languages were greatly improved with the endevel of computer size greatly improved with the endevel of computer, and computer yearly leads to more programming to take advantage of larger and faster computers. He advent of computer, sake and year on the endry 1960's, ear in the midst of a third major modifi- cation to computer usage: the improvement of tensitute, chaired by I. Teager, which examined the long range computational needs of the native, and a subsequent MIT computer working committee, chaired by I. McCarthy. Newever, the views and conclusions expressed in this paper should be taken as solely those of the present authors.72To solve these interaction process making use of the computer at the same time. The first aris or one can mean scine. The first soriented towards hardware efficiency in the sense of attemport at the same time. The first aris or one can eman scine. The first soriented towards hardware efficiency in the sense of attemport at the same time. The first aris or one can eman scine or different taks, or one can eman using different parts of the hardware efficiency in the sense of attemport at the same time. The first aris on one can eman using different parts of the hardware efficiency in the sense of attemport at the same time. The first aris on one can eman actime there, is primarily concerned wit			
Tit is the purpose of this paper to discuss briefly the need for time-sharing, some of the implementation problems, an experimental time- sharing system which has been developed for the contemportry IMM 7090, and finally a scheduling algorithm of one of us (RJC) that illustrates some of the techniques which may be employed to chance and be analyzed for the performance limits of such a time-sharing system.The motivation for time-shared computer is. This rate has changed for the last decade of widespread computer use. '00IntroductionIn part, this effect has been due to the for the active prosent authors of normal proved with the advent of computers in the mid-1950's, the problems solved were largely in the construction and maintenance of hardware; in the mid-1950's, the the advent of computer use the last of a third major modifi- tame-sharing.In part, this effect has been due to the fat that as elementary problems become mast- ered on the computer, more complex problems to take dvantage of larger and more computers. As a result, larger and more complex problems being to take dvantage of larger and more computers. the use hanguages were greatly improved with the advent of computers in the mid-1950's, the process intervition by a process called time-sharing.In part, this effect has been developed to take advantage of larger and more computers. As a result, larger and more computers, the nonitor main dimension of the time-sharing philosophy, expressed in this paper, has been developed in transitive, chaired by J. KChartly. Nuch of the time-sharing philosophy, expressed in this paper, has been developed to the last decade of wick spread sciousity by the gor consult tee, chaired by J. KChartly. Interview process interview process maing different papers, with is meeting of the consulter string, is ori	Summary		
<ul> <li>The last line sharing system.</li> <li>usage arises out of the solution and computer inter- ance any system which has been developed for the contemporary IEM (7950, and finally a scheduling algorithm of one of us (F20) that illustrates some of the techniques which may be employed to enhance and be analyzed for the performance limits of such a time-sharing system.</li> <li>Introduction</li> <li>The last dozen years of computer usage have seen great strides. In the early 1950's, the oral manusched were largely in the construction and maintenance of hardware; in the mid-1950's, the usage languages were greatly improved with the advent of computers, now in the early 1960's, the usage languages were greatly improved with the advent of computers, has been developed in same-machine, interaction by a process called time-sharing.</li> <li>Wach of the time-sharing philosophy, expressed in this paper, has been developed in this paper should be taken as solely those of the programs conclusions expressed in this paper should be taken as solely those of the programs using different parts of the hardware at the same time for different parking committee, chaired by J. McCarthy, lowever, the views and conclusions expressed in this paper should be taken as solely those of the programs tradition programming to sole computer without large economic loss and also to make each interaction provement ad takes, or one can mean sting different parts of the hardware at the same time. The first maring, often called multiprogramming, is oriented towards hardware efficiency of persons trying to use a computer "". The second maning of the elempting to align complete utili- zation of all components" "". The second maning of the esprent with the efficiency of persons trying to use a computer "". The solut only in the perspective of the total system utility.</li> </ul>			
<ul> <li>implementation problems, an experimental time- sharing system which has been developed for the contemporty TRM 7080, and finally a scheduling algorithm of one of us (FCC) that illustrates some of the techniques which may be employed to enhance and be analyzed for the performance limits of such a time-sharing system.</li> <li>Introduction</li> <li>Introduction</li> <li>The last dozen years of computer usage have seen great strides. In the early 1950's, the problems solved were largely in the construction and maintennee of hardware; in the early 1950's, the advent of computers, now in the early 1950's, the advent of computers, now in the early 1950's, the advent of computers, into in the early 1950's, the advent of computers, into in the early 1960's, the advent of computers, into in the early 1960's, the advent of computers, into in the early 1960's, the usage languages were greatly improved with early into the time-sharing philosophy, expressed in this paper, has been developed in conjunction with the work of an MIT preliminary.</li> <li>We computer, and a subsequent MIT computer struct, and a subsequent MIT computer working computer usage limiter, this best to give a more precise interpretation to time- sharing. One can mean using different passing of the computer is the same time for different tasks, or one can mean subged ifficiency in the sense of attempting to attach process making uses of the computer at the same time for different tasks, or one can mean subged ifficiency in the sense of attempting to using, "</li></ul>		-	
<ul> <li>and the second property ISM 7000, and finally a scheduling algorithm of one of us (FRC) that illustrates one and techniques which may be employed to enhance and be analyzed for the performance limits of such a time-sharing system.</li> <li>Introduction</li> <li>The last dozen years of computer usage have seen great strides. In the early 190's, the usage languages were largely in the construction and maintenance of hardware; in the mixed 90's, the usage languages were greatly limits 190's, the usage languages were greatly limits 190's, the usage languages were greatly in the construction and maintenance of hardware; in the mixed 90's, the usage languages were greatly 190's, the usage languages were greatly 190's, the usage languages were greatly 190's, the sponse strict 100 computers, now in the early 1960's, the usage languages were greatly 190's, the sponse strict 100 computers, are everal hours to a hour major modification to computer, hardware; the mixed of a third major modificative presently available is for the programme of a theore major of the strenger, which serves all dimension of the inse-sharing philosophy, expressed in this paper, has been developed in conjunction with the work of an NIT preliminary study committee, chaired by J. McCarthy, However, the views and conclusions expressed in this paper should be taken as solely those of the present authors.</li> <li>Before proceeding further, it is best to give a more precise interpretation to time-sharing, which is mean there for different parts of one can mean using different parts of the activity of others using the programme, the activity of the second meaning of thanely to the total system with a recomputer with the efficiency of present sufficiency should still be considered but on grane computer strice of the total system with a recomputer made simultaneously to the programme of the meaning of the computer strice of the total system with a recomputer made simultaneously to the state of the present match as computer at the</li></ul>		-	
<ul> <li>anothem of one of us (F2C) that illustrates one of the techniques which may be employed to enhance and be analyzed for the performance limits of such a time-sharing system.</li> <li>Introduction</li> <li>Introduction</li> <li>The last dozen years of computer usage have seen great strides. In the early 1950's, the problems solved were largely in the construction and maintenace of hardware; in the mid-1950's, the advantage of computer usage: the inprovement of man-machine interaction by a process called time-sharing.</li> <li>Nuch of the time-sharing philosophy, expressed in this paper, has been developed in conjuter use, chaired by J. Kearthy.</li> <li>Before proceeding further, it is bost to give a more precise interpretation to time-sharing.</li> <li>Before proceeding further, it is bost to give a more precise interpretation to time-sharing, which is mean solely those of the present authors.</li> <li>Before proceeding further, it is best to give a more precise interpretation to time-sharing, which is mean solely those of the present authors.</li> <li>Before proceeding further, it is best to give a more precise interpretation to time-sharing, which is mean solely those of the present authors.</li> <li>Before proceeding further, it is best to give a more precise interpretation to time-sharing, which is mean solely those of the present authors.</li> <li>Before proceeding further, it is best to give a more precise interpretation to time-sharing, which is mean the for different parts of the hardware at the same time for different parts of the netware at the same time. The first meaning, of the ally programming, is oriented towards hardware efficiency in the sense of attempting to usace to computer "". The second meaning of the present "". The second meaning of the present "". The second meaning the soft second meaning of the sense of a theory at the same time. The first were proper.</li> <li>The second meaning theore and the set precise the theory of</li></ul>	-		
<ul> <li>The section which may be employed to enhance and be analyzed for the performance limits of such a time-sharing system.</li> <li>Introduction</li> <li>The last dozen years of computer usage have seen great strides. In the early 1950's, the usage languages were greatly improved with the advent of compilers; now in the early 1950's, the usage languages were greatly improved with the advent of compilers; now in the early 1950's, the usage languages were greatly improved with the advent of compilers; now in the early 1950's, the usage languages were greatly improved with the advent of compilers; now in the early 1950's, the usage languages were greatly improved with the advent of compilers; now in the early 1950's, the second man-machine interaction by a process called time-sharing.</li> <li>Much of the time-sharing philosophy, expressed in this paper, has been developed in this paper should be taken as solely those of the present authors.</li> <li>Before proceeding further, it is best to give a more precise interpretation to time-sharing, often called multiprogramming, is oriented towards hardware efficiency in the sense of attempting to atting complete utilization of all components ''.'.'.''.'''.'''''''''''''''''''''''</li></ul>		little (and has become worse in some cases) in	
enhance and be analyzed for the performance limits of such a time-sharing system. In part, this effect has been due to the for decomposition of the computer usage have seen great strides. In the early 1950's, the problems solved were largely in the construction and maintenance of hardware; in the mid-1950's, the usage languages were greatly improved with the advent of compilers; now in the early 1950's, we are in the midst of a third major modifi- cation to computer usage: the improvement of mam-machine interaction by a process called time-sharing. Much of the time-sharing philosophy, expressed in this paper, has been developed in conjunction with the work of an MIT preliminary study committee, chaired by J. McCarthy, However, the views and conclusions expressed in this paper should be taken as solely to the hardware at the same time. The first meaning, often called multiprogramming, is soriented towards are different parts of the hardware at the same time. The first meaning, often called multiprogramming, is oriented towards are different parts of the hardware at the same time. The first meaning, often called multiprogramming, is oriented towards are different parts of the hardware at the same time. The first meaning, often called multiprogramming, is oriented towards are different parts of the hardware at the same time. The first meaning, often called multiprogramming, is oriented towards are difficent parts of the hardware at the same time. The first meaning, often called multiprogramming, is oriented towards are difficent parts the advent of time-sharing, which is meant here, persons trying to use a computer "'.'.'.'.'.'.'.'.'.'.'.'.'.'.'.'.'.'.'.		the last decade of widespread computer use. $^{10}$	78
<pre>limits of such a time-sharing system. Inits of such a time-sharing system. Introduction Introduction Interaction be advented of the dearly 1950's, the problems solved were largely in the construction and maintenace of hardware; in the mid-1950's, the problems solved were largely in the construction and maintenace of hardware; in the mid-1950's, the the advent of compliers; now in the mid-1950's, the usage languages were greatly improved with the advent of compliers; now in the early 1960's, the usage languages were greatly improved with the advent of compliers; now in the early 1960's, the usage languages were greatly improved with the advent of compliers; now in the early 1960's, the advent of compliers; now in the early 1960's, the advent of compliers; now in the early 1960's, the advent of compliers; now in the early 1960's, the advent of compliers; now in the early 1960's, the advent of compliers; now in the early 1960's, the advent of computer since; chaired by a process called time-sharing. Nuch of the time-sharing philosophy, expressed in this paper, has been developed in conjunction with the work of an MIT preliminary study committee, chaired by J. McCarthy. Nowever, the views and conclusions expressed in this paper should be taken as solely those of the present authors. Before proceeding further, it is best to give a more precise interpretation to time- sharing. One can mean several persons making use of the computer at the same time. The first meaning, often called multiprogramming, is oriented towards hardware efficiency in the sense of attempting to use a computer'.'.'.'. Computer efficiency should still be considered ture utility. </pre>		In next this offset has been due to the	79
Introductionered on the computer, more complex problems82IntroductionThe last dozen years of computer usage have seen great strides. In the early 1950's, the problems solved were largely in the construction and maintenance of hardware; in the mid-1950's, the usage languages were greatly improved with the advent of computer; now in the early 1960's, we are in the midst of a third major modifi- time-sharing.Intermative presently available is for the orgrammer to attempt to debug directly at the computer is the console, there are usually lacking the sophisticated query and persons triding committee, chaired by J. MCCarthy. However, the views and conclusions expressed in this paper should be taken as solely those of the present authors.Intermative presently available. give a more proceeding further, it is best to give a more procese interpretation to time- sharing, often called multiprogramming, is oriented towards hardware efficiency should still be considered but only in the perspective of the total system utoin of all componers '''''. The second meaning of time-sharin	•	- ,	
IntroductionImages and more complicately become to functions. As a rewritten83The last dozen years of computer usage have seen great strides. In the early 1950's, the problems solved were largely in the construction and maintenance of hardware; in the mid-1950's, the usage languages were greatly improved with the midst of a third major modifi- cation to computer usage: the improvement of man-machine interaction by a process called time-sharing.1113131313		•••	
The last dozen years of computer usage have seen great strides. In the early 1950's, the problems solved were largely in the construction and maintenance of hardware; in the mid-1950's, the usage languages were greatly improved with the advent of compilers; now in the early 1960's, we are in the midds of a third major modifi- cation to computer usage: the improvement of man-machine interaction by a process called time-sharing. Much of the time-sharing philosophy, expressed in this paper, has been developed in conjunction with the work of an MIT preliminary study committee, chaired by H. Teager, which the long range computerly that the programmer to attempt to debug directly at the computer, a process which are vitally necessary gr to allow effective interaction. Thus, what is gream and anone programmer and the conjunction with the work of an MIT preliminary study committee, chaired by H. Teager, which the Institute, and a subsequent MIT computer working committee, chaired by J. McCarthy. However, the views and conclusions expressed in this paper should be taken as solely those sharing. Often called multiprogramming, is oriented towards hardware efficiency in the sense of attempting to aging computer utili- zation of all components''.'', The second meaning of time-sharing, which is meant here, is primarily concerned with the efficiency of the resons trying to use a computer'.'.'.'.'. Computer efficiency should still be considered but only in the perspective of the total system utility. The second time, the total system utility. The second time is a sole word have the ficiency in the system. This console could as a minimum be maining of time-sharing, which is meant here, is primarily concerned with the efficiency is the system. This console could as a minimum be maining of time-sharing, which is meant here, is primarily concerned with the efficiency is the system. This console could as a minimum be maining of time-sharing, which is meant here, is primarily concerned with the efficiency is the considered but onl		•	
The last dozen years of computer usage have seen great strides. In the early 1950's, the problems solved were largely in the construction and maintenance of hardware; in the mid-1950's, the usage languages were greatly improved with the addent of compilers; now in the early 1960's, we are in the midst of a third major modifi- cation to computer usage: the improvement of man-machine interaction by a process called time-sharing. Much of the time-sharing philosophy, expressed in this paper, has been developed in conjunction with the work of an MIT preliminary study committee, chaired by J. NeCarthy. However, the views and conclusions expressed in this paper should be taken as solely those of the present authors. Defore proceeding further, it is best to give a more precise interpretation to time- sharing. One can mean using different parts of the hardware at the same time. The first meaning, oftice called multiprogramming, is oriented towards hardware efficiency in the sense of attempting to a tigin gomplete utili- station of all components''. The second meaning of time-sharing, which is meant here, is primarily concerned with the efficiency of persons trying to use a computer '' <sup>2</sup> .' <sup>4</sup> . Computer efficiency should still be considered but only in the perspective of the total system utility.	Introduction	• • • •	
<pre>seen great strides. In the early 1950's, the problems solved were largely in the construction and maintenance of hardware; in the mid-1950's, the usage languages were greatly improved with the advent of compilers; now in the early 1960's, we are in the midst of a third major modifi- cation to computer usage: the improvement of man-machine interaction by a process called time-sharing. Much of the time-sharing philosophy, expressed in this paper, has been developed in conjunction with the work of an MIT preliminary study committee, chaired by J. McCarthy. However, the views and conclusions expressed in this paper should be taken as solely those of the present authors. Defore proceeding further, it is best to give a more precise interpretation to time- sharing. One can mean several persons making use of the computer at the same time. The first meaning of time-sharing, which is meant here, is primarily concerned with the sfliciency of persons trying to use a computer'<sup>1/2, 2/4</sup>. Computer efficiency should still be considered but only in the perspective of the total system utility. Sub computer proper. Sub computer string which are string which are without large economic lass and also to make each interaction more meaningful by system. This console computer and esimultaneously available to many users in a manner somewhat for sub to use a console a this own pace and with system. This console coll das a minimum be merely a typewriter but more ideally would for the computer proper. Sub to use a console this own pace and with system. This console could as a minimum be more objective proper. Sub to use a console this own pace and with system. This console could as a minimum be more objective proper. Sub to use a console this own pace and with system. This console could as a minimum be more objective proper. Sub to use a console this own pace and with system. This console could as a minimum be more objective proper. Sub to use a console this own pace and with system. This console could as a minimum be system.</pre>	The last dozen years of computer usage have	· · ·	•
problems solved were largely in the construction and maintenance of hardware; in the mid-1950's, the usage languages were greatly improved with the advent of compilers; now in the early 1950's, we are in the midst of a third major modifi- cation to computer usage: the improvement of man-machine interaction by a process called time-sharing. Much of the time-sharing philosophy, expressed in this paper, has been developed in conjunction with the work of an MIT preliminary study committee, chaired by H. Teager, which examined the long range computational needs of the Institute, and a subsequent MIT computer working committee, chaired by J. McCarthy. Nowever, the views and conclusions expressed of the present authors. Before proceeding further, it is best to give a more precise interpretation to time- sharing. One can mean using different parts of the hardware at the same time. The first meaning, often called multiprogramming is oriented towards hardware efficiency in the sense of attempting to a <u>154, Somplete</u> utili- zation of all components <sup>1, 1, 2</sup> . The second meaning of time-sharing, which is meant here, is primarily concerned with the efficiency of persons trying to use a computer <sup>1, 2, 3, 4</sup> . Computer efficiency should still be considered but only in the perspective of the total system utility.	······································		
the usage languages were greatly improved with the advent of compilers; now in the early 1960's, we are in the midst of a third major modifi- cation to computer usage: the improvement of man-machine interaction by a process called time-sharing. Much of the time-sharing philosophy, expressed in this paper, has been developed in conjunction with the work of an MIT preliminary study committee, chaired by I. Reager, which examined the long range computational needs of the Institute, and a subsequent MIT computer working committee, chaired by J. McCarthy. However, the views and conclusions expressed in this paper should be taken as solely those of the present authors. Before proceeding further, it is best to give a more precise interpretation to time- sharing. One can mean using different tasks, or one can mean several persons making use of the computer at the same time. The first tasks, or one can mean several persons making use of the computers at the same time. The first ration of all components''.'.'. The second meaning of time-sharing, which is meant here, is primarily concerned with the efficiency of persons trying to use a computer''.'.'. Computer efficiency should still be considered but only in the perspective of the total system utility.			
<ul> <li>the advent of compilers; now in the early 1960's, we are in the midst of a third major modification to computer usage: the improvement of man-machine interaction by a process called time-sharing.</li> <li>Much of the time-sharing philosophy, expressed in this paper, has been developed in conjunction with the work of an MIT preliminary study committee, chaired by H. Teager, which examined the long range computational needs of the Institute, and a subsequent MIT computer without large economic loss and also the programs which are vitally necessary growthic is to drastically increase the rate of get examined the long. The solely those of the present authors.</li> <li>Before proceeding further, it is best to give a more precise interpretation to time-sharing. One can mean using different parts of the hardware at the same time for different tasks, or one can mean several persons making use of the computer s<sup>9</sup>.<sup>9</sup>.<sup>9</sup>.<sup>9</sup>. The second meaning of time-sharing, which is meant here, is primarily concerned with the efficiency of persons trying to use a computer<sup>9</sup>.<sup>9</sup>.<sup>9</sup>.<sup>9</sup>.<sup>9</sup>.</li> <li>Computer efficiency should still be considered but only in the perspective of the total system utility.</li> </ul>		- , - , - ,	88
<pre>we are in the midst of a third major modifi- cation to computer usage: the improvement of man-machine interaction by a process called time-sharing. Much of the time-sharing philosophy, expressed in this paper, has been developed in conjunction with the work of an MIT preliminary study committee, chaired by H. Teager, which examined the long range computational needs of the Institute, and a subsequent MIT computer working committee, chaired by J. McCarthy. However, the views and conclusions expressed in this paper should be taken as solely those of the present authors. Before proceeding further, it is best to give a more precise interpretation to time- sharing. One can mean using different tasks, or one can mean several persons making sharing, often called multiprogramming, is oriented towards hardware efficiency in the sense of attempting to attail complete utili- zation of all components''.''.'. The second meaning of time-sharing, which is meant here, is primarily concerned with the efficiency of persons trying to use a computer''.'.'.'. The second meaning of time-sharing, which is meant here, is primarily concerned with the efficiency of persons trying to use a computer''.'.'.'. The second meaning of time-sharing, which is meant here, is primarily concerned with the efficiency of persons trying to use a computer''.'.'.'. Computer efficiency should still be considered but only in the perspective of the total system utility.</pre>			
Califor to computer large, and process calledtime-sharing.Much of the time-sharing philosophy,expressed in this paper, has been developed inconjunction with the work of an MIT preliminarystudy committee, chaired by H. Teager, whichexamined the long range computational needs ofthe Institute, and a subsequent MIT computerworking committee, chaired by J. McCarthy.However, the views and conclusions expressedin this paper should be taken as solely thoseof the present authors.Before proceeding further, it is best togive a more precise interpretation to time-sharing. One can mean several persons makinguse of the computer at the same time for differenttasks, or one can mean several persons makinguse of the computer at the same time for differenttasks, or one can mean several persons makingmeaning, often called multiprogramming, isoriented towards hardware efficiency in thesense of altempting to a traing, which is meant here,is primarily concerned with the efficiency ofpersons trying to use a computer '.'.'.'.'.persons trying to use a computer '.'.'.'.'.'.'.'.'.'.'.'.'.'.'.'.''.'.'.'			
Much of the time-sharing without a problem with 19 grows		programmer to attempt to debug directly at the	
Much of the time-sharing philosophy, expressed in this paper, has been developed in conjunction with the work of an MIT preliminary study committee, chaired by H. Teager, which examined the long range computational needs of the Institute, and a subsequent MIT computer working committee, chaired by J. McCarthy. However, the views and conclusions expressed in this paper should be taken as solely those of the present authors. Before proceeding further, it is best to give a more precise interpretation to time- sharing. One can mean using different parts of the hardware at the same time. The first meaning, often called multiprogramming, is oriented towards hardware efficiency in the sense of attempting to ajtaip complete utili- zation of all components <sup>5, 9, 7, 8</sup> . The second meaning of time-sharing, which is meant here, is primarily concerned with the efficiency of persons trying to use a computer <sup>1, 2, 3, 4</sup> . Computer efficiency should still be considered but only in the perspective of the total system utility.			
Nuch of the time-sharing philosophy, expressed in this paper, has been developed in conjunction with the work of an MIT preliminary study committee, chaired by H. Teager, which examined the long range computational needs of the Institute, and a subsequent MIT computer working committee, chaired by J. McCarthy. However, the views and conclusions expressed in this paper should be taken as solely those of the present authors.Even if a typewriter is the console, there are usually lacking the sophisticated query and response programs which are vitally necessary to allow effective interaction. Thus, what is get a computer without large economic loss and also to make each interaction more meaningful by extensive and complex system programming to assist in the man-computer communication.95To solve these interaction problems we would like to have a computer made simultaneously available to many users in a manner somewhat like a telephone exchange. Each user would be able to use a console at his own pace and with- use of the computer at the same time. The first meaning, often called multiprogramming, is oriented towards hardware efficiency in the sense of attempting to attain complete utili- zation of all components'.", ". The second meaning of time-sharing, which is meant here, is primarily concerned with the efficiency of persons trying to use a computer '.'.'.'.'.'.'.'.'.'.'.'.'.'.'.'.'.'.'.	time-sharing.		
expressed in this paper, has been developed in conjunction with the work of an MIT preliminary study committee, chaired by H. Teager, which examined the long range computational needs of the Institute, and a subsequent MIT computer working committee, chaired by J. McCarthy. However, the views and conclusions expressed in this paper should be taken as solely those of the present authors. Before proceeding further, it is best to give a more precise interpretation to time- sharing. One can mean using different parts of the hardware at the same time for different meaning, often called multiprogramming, is oriented towards hardware efficiency in the sense of attempting to attain gomplete utili- zation of all components <sup>(7)</sup> , The second meaning of time-sharing, which is meant here, is primarily concerned with the efficiency of persons trying to use a computer <sup>1,2</sup> , <sup>3,4</sup> . Computer efficiency should still be considered but only in the perspective of the total system utility.	Much of the time-sharing philosophy,	-	
study committee, chaired by H. Teager, which examined the long range computational needs of the Institute, and a subsequent MIT computer working committee, chaired by J. McCarthy. However, the views and conclusions expressed in this paper should be taken as solely those of the present authors. Before proceeding further, it is best to give a more precise interpretation to time- sharing. One can mean using different parts of the hardware at the same time for different tasks, or one can mean several persons making use of the computer at the same time. The first meaning, often called multiprogramming, is oriented towards hardware efficiency in the sense of attempting to attain computer utili- zation of all components <sup>5,0,7,5</sup> . The second meaning of time-sharing, which is meant here, is primarily concerned with the efficiency of persons trying to use a computer <sup>1,2,2,4</sup> . Computer efficiency should still be considered but only in the perspective of the total system utility.		usually lacking the sophisticated query and	96
examined the long range computational needs of the Institute, and a subsequent MIT computer working committee, chaired by J. McCarthy. However, the views and conclusions expressed in this paper should be taken as solely those of the present authors. Before proceeding further, it is best to give a more precise interpretation to time- sharing. One can mean using different parts of the hardware at the same time for different tasks, or one can mean several persons making use of the computer at the same time. The first meaning, often called multiprogramming, is oriented towards hardware efficiency in the sense of attempting to attain complete utili- zation of all components <sup>5, 9, 7, 8</sup> . The second meaning of time-sharing, which is meant here, is primarily concerned with the efficiency of persons trying to use a computer 1, 2, 3, 4. Computer efficiency should still be considered but only in the perspective of the total system utility.			97
the Institute, and a subsequent MIT computer working committee, chaired by J. McCarthy. However, the views and conclusions expressed in this paper should be taken as solely those of the present authors. Before proceeding further, it is best to give a more precise interpretation to time- sharing. One can mean using different parts of the hardware at the same time for different tasks, or one can mean several persons making use of the computer at the same time. The first meaning, often called multiprogramming, is oriented towards hardware efficiency in the sense of attempting to attain complete utili- zation of all components <sup>(7, 7, 8)</sup> . The second meaning of time-sharing, which is meant here, is primarily concerned with the efficiency of persons trying to use a computer $1^{1,2,3,4}$ . Computer efficiency should still be considered but only in the perspective of the total system utility.		•	
Working commuttee, charted by 5. metatury.To solve they without farge commute tors and alsoHowever, the views and conclusions expressedto make each interaction more meaningful by101However, the views and conclusions expressedto make each interaction more meaningful by102of the present authors.assist in the man-computer communication.103Before proceeding further, it is best toTo solve these interaction problems we104give a more precise interpretation to time-to solve these interaction problems we104sharing. One can mean using different parts ofthe hardware at the same time for differenttasks, or one can mean several persons making104use of the computer at the same time. The firstto solve these interaction problems we105oriented towards hardware efficiency in thesense of attempting to attain complete utili-sustaining display. In any case, data trans-113is primarily concerned with the efficiency ofpersons trying to use a computer <sup>1,2,3,4</sup> .to major obstacle to have remote installation116computer efficiency should still be consideredto major obstacle to have remote installation116tuility.118118	·	•	
<ul> <li>Now well, the views and contributions expressed in this paper should be taken as solely those of the present authors.</li> <li>Before proceeding further, it is best to give a more precise interpretation to time-sharing. One can mean using different parts of the hardware at the same time for different tasks, or one can mean several persons making use of the computer at the same time. The first meaning, often called multiprogramming, is oriented towards hardware efficiency in the sense of attempting to attain complete utilization of all components<sup>7, 6, 7, 8</sup>. The second meaning of time-sharing, which is meant here, is primarily concerned with the efficiency of persons trying to use a computer<sup>1, 2, 3, 4</sup>.</li> <li>Computer efficiency should still be considered but only in the perspective of the total system utility.</li> <li>In this paper should be taken as solely those of the total system utility.</li> <li>In this paper should be taken as solely those of the total system utility.</li> <li>In this paper should be taken as solely those of the total system utility.</li> <li>In this paper should be taken as solely those of the total system utility.</li> </ul>	•		
of the present authors. Before proceeding further, it is best to give a more precise interpretation to time- sharing. One can mean using different parts of the hardware at the same time for different tasks, or one can mean several persons making use of the computer at the same time. The first meaning, often called multiprogramming, is oriented towards hardware efficiency in the sense of attempting to attain complete utili- zation of all components <sup>5,10,1,8</sup> . The second meaning of time-sharing, which is meant here, is primarily concerned with the efficiency of persons trying to use a computer <sup>1,2,3,4</sup> . Computer efficiency should still be considered but only in the perspective of the total system utility.			
Before proceeding further, it is best to give a more precise interpretation to time- sharing. One can mean using different parts of the hardware at the same time for different tasks, or one can mean several persons making use of the computer at the same time. The first meaning, often called multiprogramming, is oriented towards hardware efficiency in the sense of attempting to attain complete utili- zation of all components $5^{+}, 6^{+}, 8^{-}$ . The second meaning of time-sharing, which is meant here, is primarily concerned with the efficiency of persons trying to use a computer $1, 2, 3, 4$ . Computer efficiency should still be considered but only in the perspective of the total system utility.			
Before proceeding further, it is best to give a more precise interpretation to time- sharing. One can mean using different parts of the hardware at the same time for different tasks, or one can mean several persons making use of the computer at the same time. The first meaning, often called multiprogramming, is oriented towards hardware efficiency in the sense of attempting to attain complete utili- zation of all components <sup>5,6,7,8</sup> . The second meaning of time-sharing, which is meant here, is primarily concerned with the efficiency of persons trying to use a computer <sup>1,2,3,4</sup> . Computer efficiency should still be considered but only in the perspective of the total system utility.			
sharing. One can mean using different parts of the hardware at the same time for different tasks, or one can mean several persons making use of the computer at the same time. The first meaning, often called multiprogramming, is oriented towards hardware efficiency in the sense of attempting to attain complete utili- zation of all components <sup>5,6,7,8</sup> . The second meaning of time-sharing, which is meant here, is primarily concerned with the efficiency of persons trying to use a computer $1, 2, 3, 4$ . Computer efficiency should still be considered but only in the perspective of the total system utility. 100 available to many users in a manner somewhat like a telephone exchange. Each user would be able to use a console at his own pace and with- out concern for the activity of others using the system. This console could as a minimum be sustaining display. In any case, data trans- mission requirements should be such that it would 114 be no major obstacle to have remote installation from the computer proper. 116		-	
the hardware at the same time for different tasks, or one can mean several persons making use of the computer at the same time. The first meaning, often called multiprogramming, is oriented towards hardware efficiency in the sense of attempting to attain complete utili- zation of all components <sup>5,6,7,8</sup> . The second meaning of time-sharing, which is meant here, is primarily concerned with the efficiency of persons trying to use a computer <sup>1,2,3,4</sup> . Computer efficiency should still be considered but only in the perspective of the total system utility.	<b>.</b> .	-	<b>y</b> 106
tasks, or one can mean several persons making use of the computer at the same time. The first meaning, often called multiprogramming, is oriented towards hardware efficiency in the sense of attempting to attain complete utili- zation of all components <sup>5,6,7,8</sup> . The second meaning of time-sharing, which is meant here, is primarily concerned with the efficiency of persons trying to use a computer <sup>1,2,3,4</sup> . Computer efficiency should still be considered but only in the perspective of the total system utility.		-	
use of the computer at the same time. The first meaning, often called multiprogramming, is oriented towards hardware efficiency in the sense of attempting to attain complete utili- zation of all components <sup>5,6,7,8</sup> . The second meaning of time-sharing, which is meant here, is primarily concerned with the efficiency of persons trying to use a computer <sup>1,2,3,4</sup> . Computer efficiency should still be considered but only in the perspective of the total system utility. Use of the computer at the same time. The first out concern for the activity of others using the system. This console could as a minimum be merely a typewriter but more ideally would to ntain an incrementally modifiable self- sustaining display. In any case, data trans- mission requirements should be such that it would the no major obstacle to have remote installation from the computer proper. 116 117 118			
oriented towards hardware efficiency in the sense of attempting to attain complete utili- zation of all components <sup>5,6,7,8</sup> . The second meaning of time-sharing, which is meant here, is primarily concerned with the efficiency of persons trying to use a computer <sup>1,2,3,4</sup> . Computer efficiency should still be considered but only in the perspective of the total system utility. Induction of all combols of the total system utility. Induction of all combols of the total system incomplete utili- contain an incrementally modifiable self- sustaining display. In any case, data trans- mission requirements should be such that it would 114 be no major obstacle to have remote installation 115 from the computer proper. 116	•	out concern for the activity of others using the	
sense of attempting to attain complete utili- zation of all components <sup>5,6,7,8</sup> . The second meaning of time-sharing, which is meant here, is primarily concerned with the efficiency of persons trying to use a computer <sup>1,2,3,4</sup> . Computer efficiency should still be considered but only in the perspective of the total system utility. Ontain an incrementally modifiable self- sustaining display. In any case, data trans- mission requirements should be such that it would the no major obstacle to have remote installation from the computer proper. 112 sustaining display. In any case, data trans- mission requirements should be such that it would 114 be no major obstacle to have remote installation 115 from the computer proper. 116 117 118		-	
meaning of time-sharing, which is meant here, is primarily concerned with the efficiency of persons trying to use a computer 1,2,3,4. Computer efficiency should still be considered but only in the perspective of the total system utility. mission requirements should be such that it would 114 be no major obstacle to have remote installation 115 from the computer proper. 116 117 117 118 119	oriented towards hardware efficiency in the	• • •	
meaning of time-sharing, which is meant here, is primarily concerned with the efficiency of persons trying to use a computer 1,2,3,4. Computer efficiency should still be considered but only in the perspective of the total system utility. mission requirements should be such that it would 114 be no major obstacle to have remote installation 115 from the computer proper. 116 117 117 118 119	zation of all components 5,6,7,8. The second		
persons trying to use a computer 1,2,3,4from the computer proper.116Computer efficiency should still be considered117but only in the perspective of the total system118utility.119			-
Computer efficiency should still be considered110but only in the perspective of the total system117utility.118	is primarily concerned with the efficiency of 1.2.3.4		115
but only in the perspective of the total system 117 utility. 119		from the computer proper.	
utility. 118 119			

1 The basic technique for a time-sharing system 2 is to have many persons simultaneously using the З computer through typewriter consoles with a time-sharing supervisor program sequentially 4 running each user program in a short burst or 5 quantum of computation. This sequence, which in 6 the most straightforward case is a simple round-7 robin, should occur often enough so that each 8 user program which is kept in the high-speed 9 memory is run for a quantum at least once during 10 each approximate human reaction time ( $\sim$ .2 11 seconds). In this way, each user sees a computer fully responsive to even single key strokes each 12 of which may require only trivial computation; 13 in the non-trivial cases, the user sees a 14 gradual reduction of the response time which is 15 proportional to the complexity of the response 16 calculation, the slowness of the computer, and 17 the total number of active users. It should be 18 clear, however, that if there are n users 19 actively requesting service at one time, each 20 user will only see on the average 1/n of the 21 effective computer speed. During the period of high interaction rates while debugging programs, 22 this should not be a hindrance since ordinarily 23 the required amount of computation needed for 24 each debugging computer response is small 25 compared to the ultimate production need. 26 27 Not only would such a time-sharing system 28 improve the ability to program in the conventional 29 manner by one or two orders of magnitude, but there would be opened up several new forms of 30 computer usage. There would be a gradual 31 reformulation of many scientific and engineering 32 applications so that programs containing decision 33 trees which currently must be specified in 34 advance would be eliminated and instead the 35 particular decision branches would be specified 36 only as needed. Another important area is that 37 of teaching machines which, although frequently 38 trivial computationally, could naturally 39 exploit the consoles of a time-sharing system with the additional bonus that more elaborate 40 and adaptive teaching programs could be used. 41 Finally, as attested by the many small business 42 computers, there are numerous applications in 43 busines and in industry where it would be 44 advantageous to have powerful computing facilities 45 available at isolated locations with only the 46 incremental capital investment of each console. 47 But it is important to realize that even without the above and other new applications, the major 48 advance in programming intimacy available from 49 time-sharing would be of immediate value to 50 computer installations in universities, research 51 laboratories, and engineering firms where 52 program debugging is a major problem. 53 54 55 56

- 57
- 58
- 59
- 60

## PROGRAMMING AND CODING

Implementation Problems	61
	62
As indicated, a straightforward plan for	63
time-sharing is to execute user programs for	64
small quantums of computation without priority	65
in a simple round-robin; the strategy of time- sharing can be more complex as will be shown	66
later, but the above simple scheme is an	67
adequate solution. There are still many	68
problems, however, some best solved by hard-	69
ware, others affecting the programming conven-	70
tions and practices. A few of the more	71
obvious problems are summarized:	72
	73
Hardware Problems:	74
	 75
1. Different user programs if simultan-	76
eously in core memory may interfere with each	77
other or the supervisor program so some form of	78
memory protection mode should be available when	79
operating user programs.	79 80
2. The time-sharing supervisor may need	81
at different times to run a particular program	
from several locations. (Loading relocation	82 83
bits are no help since the supervisor does not	
know how to relocate the accumulator, etc.)	84 05
Dynamic relocation of all memory accesses that	85
pick up instructions or data words is one	86
effective solution.	87
	88
3. Input-output equipment may be initiated	89
by a user and read words in on another user	90
program. A way to avoid this is to trap all	91
input-output instructions issued by a user's program when operated in the memory protection	92
mode.	93
mode.	94
4. A large random-access back-up storage	95
is desirable for general program storage files	96
for all users. Present large capacity disc	97
units appear to be adequate.	98
	99
5. The time-sharing supervisor must be	100
able to interrupt a user's program after a	101
quantum of computation. A program-initiated one-	102
shot multivibrator which generates an interrupt	103
a fixed time later is adequate.	104
6. Large core memories (e.g. a million	105
at Tweele core memorres (c.2, a militude	

106 words) would ease the system programming complications immensely since the different active 107 user programs as well as the frequently used 108 system programs such as compilers, query programs, 109 etc. could remain in core memory at all times. 110

## Programming Problems:

113 1. The supervisor program must do auto-114 matic user usage charge accounting. In general,

111

112

2/10

1 2 the user should be charged on the basis of a 3 system usage formula or algorithm which should 4 include such factors as computation time, amount 5 of high-speed memory required, rent of secondary 6 memory storage, etc. 7 2. The supervisor program should coordinate 8 all user input-output since it is not desirable 9 to require a user program to remain constantly 10 in memory during input-output limited operations. 11 In addition, the supervisor must coordinate all 12 usage of the central, shared high-speed input-13 output units serving all users as well as the 14 clocks, disc units, etc. 15 16 3. The system programs available must be potent enough so that the user can think about 17 his problem and not be hampered by coding 18 details or typographical mistakes. Thus, 19 compilers, query programs, post-mortem programs, 20 loaders, and good editing programs are 21 essential. 22 23 4. As much as possible, the users should 24 be allowed the maximum programming flexibility 25 both in choices of language and in the absence 26 of restrictions. 27 Usage Problems 28 29 1. Too large a computation or excessive 30 typewriter output may be inadvertently requested 31 so that a special termination signal should be 32 available to the user. 33 34 2. Since real-time is not computer usage-35 time, the supervisor must keep each user informed so that he can use his judgment regarding loops, 36 37 etc. 38 3. Computer processor, memory and tape 39 malfunctions must be expected. Basic operational 40 questions such as "Which program is running?" 41 must be answerable and recovery procedures fully 42 anticipated. 43 44 45 An Experimental Time-Sharing System for the IBM 46 7090 47 48 Having briefly stated a desirable time-49 sharing performance, it is pertinent to ask what level of performance can be achieved with 50

existant equipment. To begin to answer this
question and to explore all the programming and
operational aspects, an experimental timesharing system has been developed. This system
was originally written for the IBM 709 but has
since been converted for use with the 7090
computer.

57

58

59 60

-

62 The 7090 of the MIT Computation Center has, 63 in addition to three channels with 19 tape units. 64 a fourth channel with the standard Direct Data 65 Connection. Attached to the Direct Data Connec-66 tion is a real-time equipment buffer and control 67 rack designed and built under the direction of H. Teager and his group. This rack has a variety 68 of devices attached but the only ones required 69 by the present systems are three flexowriter 70 typewriters. Also installed on the 7090 are two 71 special modifications (i.e. RPQ's); a standard 72 60 cycle accounting and interrupt clock, and a 73 special mode which allows memory protection, 74 dvnamic relocation and trapping of all user 75 attempts to initiate input-output instructions. 76

In the present system the time-sharing occurs between four users, three of whom are online each at a typewriter in a foreground system, and a fourth passive user of the background Fap-Mad-Madtran-BSS Monitor System similar to the Fortran-Fap-BSS Monitor System (FMS) used by most of the Center programmers and by many other 7090 installations.

Significant design features of the foreground system are:

1. It allows the user to develop programs in languages compatible with the background system,

2. Develop a private file of programs,

3. Start debugging sessions at the state of the previous session, and

4. Set his own pace with little waste of computer time.

98 Core storage is allocated such that all users 99 operate in the upper 27,000 words with the time-100 sharing supervisor (TSS) permanently in the lower 5,000 words. To avoid memory allocation 101 clashes, protect users from one another, and 102 simplify the initial 709 system organization, 103 only one user was kept in core memory at a 104 time. However, with the special memory protec-105 tion and relocation feature of the 7090, more 106 sophisticated storage allocation procedures are 107 being implemented. In any case, user swaps are 108 minimized by using 2-channel overlapped magnetic tape reading and writing of the pertinent 109 locations in the two user programs. 110

The foreground system is organized around commands that each user can give on his typewriter and the user's private program files which presently (for want of a disc unit) are kept on a separate magnetic tape for each user. 112 113 114 115 116

\* This group is presently using another approach in developing a time-sharing system for the MIT 7090. 117 118 117 118 117 118 119 120

337

61

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

111

1 For convenience the format of the private tape files is such that they are card images, have 2 title cards with name and class designators and З can be written or punched using the off-line 4 equipment. (The latter feature also offers a 5 crude form of large-scale input-output.) The 6 magnetic tape requirements of the system are the 7 seven tapes required for the normal functions of 8 the background system, a system tape for the 9 time-sharing supervisor that contains most of 10 the command programs, and a private file tape and dump tape for each of the three foreground 11 users. 12

13

The commands are typed by the user to the 14 time-sharing supervisor (not to his own program) 15 and thus can be initiated at any time regardless 16 of the particular user program in memory. For 17 similar coordination reasons, the supervisor 18 handles all input-output of the foreground 19 system typewriters. Commands are composed of segments separated by vertical strokes; the 20 first segment is the command name and the 21 remaining segments are parameters pertinent to 22 the command. Each segment consists of the last 23 6 characters typed (starting with an implicit 24 6 blanks) so that spacing is an easy way to 25 correct a typing mistake. A carriage return is 26 the signal which initiates action on the command. 27 Whenever a command is received by the supervisor, "WAIT", is typed back followed by "READY." when 28 29 the command is completed. (The computer responses are always in the opposite color from the user's 30 typing.) While typing, an incomplete command 31 line may be ignored by the "quit" sequence of a 32 code delete signal followed by a carriage return. .3.3 Similarly after a command is initiated, it may 34 be abandoned if a "quit" sequence is given. In 35 addition, during unwanted command typeouts, the 36 command and output may be terminated by pushing a special "stop output" button. 37 38

The use of the foreground system is initiated 39 whenever a typewriter user completes a command 40 line and is placed in a waiting command queue. 41 Upon completion of each quantum, the time-sharing 42 supervisor gives top priority to initiating any 43 waiting commands. The system programs corres-44 ponding to most of the commands are kept on the 45 special supervisor command system tape so that to 46 avoid waste of computer time, the supervisor continues to operate the last user program until 47 the desired command program on tape is positioned 48 for reading. At this point, the last user is 49 read out on his dump tape, the command program 50 read in, placed in a working status and initiated 51 as a new user program. However, before starting 52 the new user for a quantum of computation, the 53 supervisor again checks for any waiting command 54 of another user and if necessary begins the lookahead positioning of the command system tape 55 56 while operating the new user.

- 57
- 58
- 59
- 60

## PROGRAMMING AND CODING

Whenever the waiting command queue is 61 empty, the supervisor proceeds to execute a 62 simple round-robin of those foreground user 63 programs in the working status queue. Finally, 64 if both these queues are empty, the background 65 user program is brought in and run a quantum at 66 a time until further foreground system actively 67 develops. 68

69 Foreground user programs leave the working 70 status queue by two means. If the program proceeds to completion, it can reenter the 71 supervisor in a way which eliminates itself and 72 places the user in dead status; alternatively, 73 by a different entry the program can be placed 74 in a dormant status (or be manually placed by 75 the user executing a quit sequence). The dormant 76 status differs from the dead status in that the 77 user may still restart or examine his program. 78

User input-output is through each type-79 writer, and even though the supervisor has a 80 few lines of buffer space available, it is 81 possible to become input-output limited. 82 Consequently, there is an additional input-83 output wait status, similar to the dormant, 84 which the user is automatically placed in by 85 the supervisor program whenever input-output 86 delays develop. When buffers become near 87 empty on output or near full on input, the user program is automatically returned to the working 88 status; thus waste of computer time is avoided. 89 90

Commands

To clarify the scope of the foreground system and to indicate the basic tools available to the user, a list of the important commands follows along with brief summaries of their operations:

1. α

 $\alpha$  = arbitrary text treated as a comment. 100

- 2. login  $\alpha \beta$ 
  - $\alpha$  = user problem number 104  $\beta$  = user programmer number 105

Should be given at beginning of each user's session. Rewinds user's private file tape; clears time accounting records. 108 109

logout

Should be given at end of each user's 111 session. Rewinds user's private file tape; 112 punches on-line time accounting cards. 113 114

4. input

Sets user in input mode and initiates 116 automatic generation of line numbers. The user 117

118 119 120

115

91

92

93

94

95

96

97

98

99

102

103

110

### AN EXPERIMENTAL TIME-SHABING SYSTEM

```
types a card image per line according to a
1
     format appropriate for the programming language.
2
     (The supervisor collects these card images at
З
     the end of the user's private file tape.) When
4
     in the automatic input mode, the manual mode may
5
     be entered by giving an initial carriage return
6
     and typing the appropriate line number followed
     by | and line for as many lines as desired. To
8
     reenter the automatic mode, an initial carriage
9
     return is given.
10
          The manual mode allows the user to over-
11
     write previous lines and to insert lines. (cf.
12
     File Command.)
13
14
          5. edit \alpha \mid \beta
15
              \alpha = title of file
16
17
              \beta = class of file
18
              The user is set in the automatic input
19
     mode with the designated file treated as initial
20
     input lines. The same conventions apply as to
21
     the input command.
22
          6. file \alpha \mid \beta
23
24
              \alpha = title to be given to file
25
              \beta = class of language used during input
26
27
              The created file will consist of the
     numbered input lines (i.e. those at the end of
28
     the user's private file tape) in sequence; in
29
     the case of duplicate line numbers, the last
30
     version will be used. The line numbers will be
31
     written as sequence numbers in the corresponding
32
     card images of the file.
33
34
          For convenience the following editing
35
     conventions apply to input lines:
36
          a. an underline signifies the deletion of
37
     the previous characters of the line.
38
39
          b. a backspace signifies the deletion of
40
     the previous character in the field.
41
          The following formats apply:
42
43
          a. FAP: symbol, tab, operation, tab,
44
     variable field and comment.
45
          b. MAD, MADTRAN, FORTRAN: statement label,
46
     tab, statement. To place a character in the
47
     continuation column: statement label, tab,
48
     backspace, character, statement.
49
          c. DATA: cols. 1-72.
50
51
          7. fap \alpha
52
              Causes the file designated as \alpha, fap to
53
     be translated by the FAP translator (assembler).
54
     Files \alpha, symtb and \alpha, bss are added to the user's
55
     private file tape giving the symbol table and
56
     the relocatable binary BSS form of the file.
57
58
59
60
```

```
8. mad \alpha
```

Causes file  $\alpha$ , mad to be translated by the MAD translator (compiler). File  $\alpha$ , bss is created.

9. madtrn  $\alpha$ 

Causes file  $\alpha$ , madtrn (i.e. a pseudo-Fortran language file) to be edited into an equivalent file  $\alpha$ , mad (added to the user's file) and translation occurs as if the command mad  $\alpha$ had been given.

10. load 
$$\alpha_1 \mid \alpha_2 \mid \dots \mid \alpha_n$$

Causes the consecutive loading of files  $\alpha_i$ , bss (i=1,2,...,n). An exception occurs if  $\alpha_i =$ (libe), in which case file  $\alpha_{i+1}$ , bss is searched as a library file for all subprograms still missing. (There can be further library files.)

11. use 
$$|\alpha_1| |\alpha_2| \dots |\alpha_n|$$

This command is used whenever a load or previous use command notifies the user of an incomplete set of subprograms. Same  $\alpha_i$  conventions as for load.

start  $|\alpha|\beta$ 12.

Starts the program setup by the load and use commands (or a dormant program) after first positioning the user private file tape in front of the title card for file  $\alpha,\beta$ . (If  $\beta$  is not given, a class of data is assumed; if both  $\alpha$  and  $\beta$  are not given, no tape movement occurs and the program is started.)

13. pm α

 $\alpha$  = "lights", "stomap", or the usual format of the standard Center post-mortem (F2PM) request: subprogram name | loc<sub>1</sub> | loc<sub>2</sub> | mode | direction where mode and direction are optional.

Produces post-mortem of user's dormant program according to request specified by  $\alpha$ . (E.g. matrix | 5 | 209 | flo | rev will cause to be printed on the user's typewriter the contents of subprogram "matrix" from relative locations 5 to 209 in floating point form and in reverse sequence.)

# 14. skippm

Used if a pm command is "quit" during output and the previous program interruption is to be restarted.

#### 15. listf

Types out list of all file titles on user's private file tape.

339

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

1 16. printf  $|\alpha|\beta|\gamma$ 2 Types out file  $\alpha, \beta$  starting at line З number  $\gamma$ . If  $\gamma$  is omitted, the initial line is 4 assumed. Whenever the user's output buffer fills, 5 the command program goes into an I/O wait status 6 allowing other users to time-share until the buffer needs refilling. 7 8 17. xdump  $\alpha \mid \beta$ 9 10 Creates file  $\alpha, \beta$  (if  $\beta$  omitted, Xdump assumed) on user's private file tape consisting 11 of the complete state of the user's last dormant 12 program. 13 14 18. xdump  $\alpha \mid \beta$ 15 Inverse of xdump command in that it 16 resets file  $\alpha, \beta$  as the user's program, starting 17 it where it last left off. 18 19 Although experience with the system to date 20 is quite limited, first indications are that 21 programmers would readily use such a system if it 22 were generally available It is useful to ask, 23 now that there is some operating experience with the 7090 system, what observations can be made. 24 An immediate comment is that once a user gets 25 accustomed to computer response, delays of even 26 a fraction of a minute are exasperatingly long, 27 an effect analogous to conversing with a slow-28 speaking person. Similarly, the requirement that 29 a complete typewritten line rather than each 30 character be the minimum unit of man-computer 31 communication is an inhibiting factor in the sense that a press-to-talk radio-telephone con-32 versation is more stilted than that of an 33 ordinary telephone. Since maintaining a rapid 34 computer response on a character by character 35 basis requires at least a vestigial response 36 program in core memory at all times, the straight-37 forward solution within the present system is to 38 have more core memory available. At the very 39 least, an extra bank of memory for the time-40 sharing supervisor would ease compatibility prob-41 lems with programs already written for 32,000 word 7090's. 42 43 For reasons of expediency, the weakest 44 portions of the present system are the conventions 45 for input, editing of user files, and the degree 46 of rapid interaction and intimacy possible while 47 debugging. Since to a large extent these areas 48 involve the taste, habits, and psychology of the 49 users, it is felt that proper solutions will require considerable experimentation and prag-50 matic evaluation; it is also clear that these 51 areas cannot be treated in the abstract for the 52 programming languages used will influence greatly 53 the appropriate techniques. A greater use of 54 symbolic referencing for locations, program names 55 and variables is certainly desired; symbolic post-56 mortem programs, trace programs, and before-and-57 after differential dump programs should play 58 useful roles in the debugging procedures. 59

60

#### PROGRAMMING AND CODING

In the design of the present system, great 61 care went into making each user independent of 62 the other users. However, it would be a useful 63 extension of the system if this were not always 64 the case. In particular, when several consoles 65 are used in a computer controlled group such as 66 in management or war games, in group behavior 67 studies, or possibly in teaching machines, it 68 would be desirable to have all the consoles 69 communicating with a single program. 70

Another area for further improvement within the present system is that of file maintenance, since the presently used tape units are a hindrance to the easy deletion of user program files. Disc units will be of help in this area as well as with the problem of consolidating and scheduling large-scale central input-output generated by the many console users.

Finally, it is felt that it would be desir-79 able to have the distinction between the fore-80 ground and background systems eliminated. The 81 present-day computer operator would assume the 82 role of a stand-in for the background users, 83 using an operator console much like the other 84 user consoles in the system, mounting and de-85 mounting magnetic tapes as requested by the 86 supervisor, receiving instructions to read card 87 decks into the central disc unit, etc. Similarly the foreground user, when satisfied with his 88 program, would by means of his console and the 89 supervisor program enter his program into the 90 queue of production background work to be 91 performed. With these procedures implemented 92 the distinction of whether one is time-sharing 93 or not would vanish and the computer user would 94 be free to choose in an interchangable way that 95 mode of operation which he found more suitable 96 at a particular time.

#### A Multi-Level Scheduling Algorithm

100 Regardless of whether one has a million 101 word core memory or a 32,000 word memory as currently exists on the 7090, one is inevitably 102 faced with the problem of system saturation 103 where the total size of active user programs 104 exceeds that of the high-speed memory or there 105 are too many active user programs to maintain 106 an adequate response at each user console. 107 These conditions can easily arise with even a 108 few users if some of the user programs are 109 excessive in size or in time requirements. The predicament can be alleviated if it is assumed 110 that a good design for the system is to have a 111 saturation procedure which gives graceful degradation of the response time and effective real-time computation speed of the large and long-running users.

71

72

73

74

75

76

77

78

97

98

99

118

## AN EXPERIMENTAL TIME-SHARING SYSTEM

1 To show the general problem, Figure 1 2 qualitatively gives the user service as a function З of n, the number of active users. This service parameter might be either of the two key factors: 4 computer response time or n times the real-time 5 computation speed. In either case there is some 6 critical number of active users, N, representing 7 the effective user capacity, which causes satur-8 ation. If the strategy near saturation is to 9 execute the simple round-robin of all users, then 10 there is an abrupt collapse of service due to the 11 sudden onset of the large amount of time required 12 to swap programs in-and-out of the secondary memory such as a disc or drum unit. Of course. 13 Figure 1 is quite qualitative since it depends 14 critically on the spectrum of user program sizes 15 as well as the spectrum of user operating times. 16

To illustrate the strategy that can be employed to improve the saturation performance
of a time-sharing system, a multi-level scheduling algorithm is presented. This algorithm also
can be analyzed to give broad bounds on the
system performance.

The basis of the multi-level scheduling algorithm is to assign each user program as it enters the system to be run (or completes a response to a user) to an  $\ell$ th level priority queue. Programs are initially entered into a level  $\ell_0$ , corresponding to their size such that

 $\ell_{o} = \left[ \log_{2} \left( \left[ \frac{w_{p}}{w_{q}} \right] + 1 \right) \right]$ 

32 where  $w_{D}$  is the number of words in the program, 33 w is the number of words which can be trans-mitted in and out of the high-speed memory from 34 35 the secondary memory in the time of one quantum, 36 q, and the bracket indicates "the integral part 37 of". Ordinarily the time of a quantum, being 38 the basic time unit, should be as small as 39 possible without excessive overhead losses when 40 the supervisor switches from one program in high-41 speed memory to another. The process starts with 42 the time-sharing supervisor operating the program at the head of the lowest level occupied queue,  $\ell$ , for up to  $2^\ell$  quanta of time and then if the 43 44 program is not completed (i.e. has not made a 45 response to the user) placing it at the end of 46 the l+1 level queue. If there are no programs 47 entering the system at levels lower than l, this 48 process proceeds until the queue at level  $\ell$  is 49 exhausted; the process is then iteratively begun 50 again at level l+1, where now each program is 51 run for  $2^{l+1}$  quanta of time. If during the execution of the  $2^{\ell}$  quanta of a program at level 52 53  $\ell$ , a lower level,  $\ell$ ', becomes occupied, the current user is replaced at the head of the *l*th 54 queue and the process is reinitiated at level 55 l'. 56

57 58

30

31

59 60 Similarly, if a program of size  $w_p$  at level l, during operation requests a change n memory size from the time-sharing supervisor, then the enlarged (or reduced) version of the program should be placed at the end of the l'' queue where

$$\mathcal{P}'' = \ell + \left[ \log_2 \left( \left[ \frac{w_p''}{w_p} \right] + 1 \right) \right]$$
 (2)

Again the process is re-initiated with the headof-the-queue user at the lowest occupied level of  $\ell'$ .

Several important conclusions can be drawn from the above algorithm which allow the performance of the system to be bounded.

## Computational Efficiency

1. Because a program is always operated for a time greater than or equal to the swap time (i.e. the time required to move the program in and out of secondary memory), it follows that the computational efficiency never falls below one-half. (Clearly, this fraction is adjustable in the formula for the initial level,  $l_0$ .) An alternative way of viewing this bound is to say that the real-time computing speed available to one out of n active users is no worse than if there were 2n active users all of whose programs were in the high-speed memory.

## Response Time

(1)

2. If the maximum number of active users is N, then an individual user of a given program size can be guaranteed a response time,

$$t_{r} \leq 2Nq \left( \left[ \frac{w_{p}}{w_{q}} \right] + 1 \right)$$
 (3)

since the worst case occurs when all competing user programs are at the same level. Conversely, if  $t_r$  is a guaranteed response of arbitrary value and the largest size of program is assumed, then the maximum permissible number of active users is bounded.

Long Runs

105 3. The relative swap time on long runs can 106 be made vanishingly small. This conclusion 107 follows since the longer a program is run, the 108 higher the level number it cascades to with a 109 correspondingly smaller relative swap time. It is an important feature of the algorithm that long 110runs must in effect prove they are long so that 111 112 programs which have an unexpected demise are detected quickly. In order that there be a 113 finite number of levels, a maximum level number, 114 L, can be established such that the asymptotic 115 swap overhead is some arbitrarily small 116 percentage, p:

117 118 119

120

Note: formula (2) is incorrect. To correct, remove "+ 1" and remove inner brackets around ratio of program sizes.

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

$$\begin{array}{ccc} 1 \\ 2 \\ 3 \\ 4 \end{array} \quad \text{L = } \left[ \log_2 \left( \left[ \frac{w_{pmax}}{pw_q} \right] + 1 \right) \right] \quad (4) \\ \text{where } w_{pmax} \text{ is the size of the largest possible} \end{array}$$

program.

## Multi-level vs. Single-level Response Times

8 4. The response time for programs of equal 9 size, entering the system at the same time, and 10 being run for multiple quanta, is no worse than 11 approximately twice the response-time occurring in a single quanta round-robin procedure. If 12 there are n equal sized programs started in a 13 queue at level l, then the worst case is that of 14 the end-of-the-queue program which is ready to 15 respond at the very first quantum run at the 16 l+j level. Using the multi-level algorithm, the 17 total delay for the end-of-the-queue program is 18 by virtue of the geometric series of quanta:

$$T_m \sim q 2^{\ell} \{ n(2^{j}-1) + (n-1) 2^{j} \}$$
 (5)

21 Since the end-of-the-queue user has computed for 22 a time of  $2^{\lambda}(2^{J}-1)$  quanta, the equivalent single-23 level round-robin delay before a response is: 24

$$T_{s} \sim q^{2^{\ell}} \{ n(2^{j-1}) \}$$
 (6)

27 Hence

$$\frac{T_{\rm m}}{T_{\rm s}} \sim 1 + \left(\frac{\rm n-1}{\rm n}\right) \left(\frac{\rm 2^{\rm j}}{\rm 2^{\rm j}-1}\right) \sim 2 \qquad (7)$$

and the assertion is shown. It should be noted 32 that the above conditions, where program swap 33 times are omitted, which are pertinent when all 34 programs remain in high-speed memory, are the 35 least favorable for the multi-level algorithm; if 36 swap times are included in the above analysis, the 37 ratio of  $T_m / T_s$  can only become smaller and may 38 become much less than unity. By a similar analysis 39 it is easy to show that even in the unfavorable 40 case where there are no program swaps, head-of-the-41 queue programs that terminate just as the  $2^{\ell+j}$ quanta are completed receive under the multi-42 level algorithm a response which is twice as 43 fast as that under the single-level round-robin 44  $\overline{(i.e. T_m / T_s = 1/2)}$ . 45

### Highest Serviced Level

48 5. In the multi-level algorithm the level 49 classification procedure for programs is entirely 50 automatic, depending on performance and program size rather than on the declarations (or hopes) 51 of each user. As a user taxes the system, the 52 degradation of service occurs progressively 53 starting with the higher level users of either 54 large or long-running programs; however, at some 55 level no user programs may be run because of too 56 many active users at lower levels. To determine 57

58

46

47

- 59
- 60

61 a bound on this cut-off point we consider N active users at level l each running 2<sup>l</sup> quanta, 62 terminating, and reentering the system again at 63 level l at a user response time, t<sub>u</sub>, later. If there is to be no service at level l+1, then the 64 65 computing time, Nq2<sup> $\ell$ </sup>, must be greater than or 66 equal to t<sub>u</sub>. Thus the guaranteed active levels, 67  $l_a$ , are given by the relation: 68

$$\ell_{a} \xi \left[ \log_{2} \left( \frac{t_{u}}{Nq} \right) \right]$$
(8) 69
70
71

In the limit, t<sub>u</sub> could be as small as a minimum 72 user reaction time (~.2 sec.), but the expected 73 value would be several orders of magnitude 74 greater as a result of the statistics of a 75 large number of users. 76

77 The multi-level algorithm as formulated above makes no explicit consideration of the seek 78 or latency time required before transmission of 79 programs to and from disc or drum units when 80 they are used as the secondary memory, (although 81 formally the factor  ${\bf w}_{\bf q}$  could contain an average figure for these times). One simple modification 82 83 to the algorithm which usually avoids wasting 84 the seek or latency time is to continue to 85 operate the last user program for as many quanta 86 as are required to ready the swap of the new user with the least priority user; since ordinarily 87 only the higher level number programs would be 88 forced out into the secondary memory, the 89 extended quanta of operation of the old user 90 while seeking the new user should be but a minor 91 distortion of the basic algorithm. 92

93 Further complexities are possible when the hardware is appropriate. In computers with input- 94output channels and low transmission rates to and 95 from secondary memory, it is possible to overlap 96 the reading and writing of the new and old users 97 in and out of high-speed memory while operating 98 the current user. The effect is equivalent to 99 using a drum giving 100 % multiplexor usage but 100 there are two liabilities, namely, no individual 101 user can utilize all the available user memory 102 space and the look-ahead procedure breaks down 103 whenever an unanticipated scheduling change 104 occurs (e.g. a program terminates or a higherpriority user program is initiated). 105

106 Complexity is also possible in storage 107 allocation but certainly an elementary procedure 108 and a desirable one with a low-transmission rate 109 secondary memory is to consolidate in a single 110 block all high-priority user programs whenever 111 sufficient fragmentary unused memory space is 112 available to read in a new user program. Such a procedure is indicated in the flow diagram of the 113 multi-level scheduling algorithm which is given 114 as Figure 2. 115

120

8/10

5

6

7

19

20

25

26

28

29

30

## AN EXPERIMENTAL TIME-SHARING SYSTEM

1 It should also be noted that Figure 2 only 2 accounts for the scheduling of programs in a working status and still does not take into 3 account the storage allocation of programs which 4 are in a dormant (or input-output wait status). 5 One systematic method of handling this case is 6 to modify the scheduling algorithm so that 7 programs which become dormant at level  $\ell$  are 8 entered into the queue at level l+1. The 9 scheduling algorithm proceeds as before with the 10 dormant programs continuing to cascade but not 11 operating when they reached the head of a queue. 12 Whenever a program must be removed from highspeed memory, a program is selected from the end-13 of-the-queue of the highest occupied level 14 number. 15

Finally, it is illuminating to apply the
multi-level scheduling algorithm bounds to the
contemporary IBM 7090. The following approximate
values are obtained:

q = 16 m.s. (based on 1% switching overhead)
wq = 120 words (based on one IBM 1301 model
 2 disc unit without seek or latency
 times included)

 $t_r \leq 8Nf_{sec.}$  (based on programs of (32k)f words)

 $l_a \leq \log_2$  (1000/N) (based on t<sub>u</sub> = 16 sec.)

 $l_0 \leqslant 8$  (based on a maximum program size of 32K words)

32 Using the arbitrary criteria that programs 33 up to the maximum size of 32,000 words should 34 always get some service, which is to say that 35  $\max l_a = \max l_o$ , we deduce as a conservative 36 estimate that N can be 4 and that at worst the 37 response time for a trivial reply will be 32 38

39 The small value of N arrived at is a direct consequence of the small value of  $\boldsymbol{w}_q$  that results 40 from the slow disc word transmission rate. This 41 rate is only 3.3% of the maximum core memory 42 multiplexor rate. It is of interest that using 43 high-capacity high-speed drums of current design 44 such as in the Sage System of in the IBM Sabre 45 System it would be possible to attain nearly 46 100% multiplexor utilization and thus multiply 47 W<sub>c</sub> by a factor of 30. It immediately follows 48 that user response times equivalent to those 49 given above with the disc unit would be given 50 to 30 times as many persons or to 120 users; the 51 total computational capacity, however, would not change. 52

53

21

22

23

24

25

26

27

28

29

30

31

- 54 55
- 56
- 57
- 58
- 59
- 60

In any case, considerable caution should be 61 62 used with capacity and computer response time estimates since they are critically dependent 63 upon the distribution functions for the user 64 response time, t<sub>u</sub>, and the user program size, 65 w<sub>p</sub>, and the computational capacity requested by 66 each user. Past experience using conventional 67 programming systems is of little assistance be-68 cause these distribution functions will depend 69 very strongly upon the programming systems made 70 available to the time-sharing users as well as 71 upon the user habit patterns which will gradually evolve. 72

#### Conclusions

76 In conclusion, it is clear that contemporary 77 computers and hardware are sufficient to allow 78 moderate performance time-sharing for a limited 79 number of users. There are several problems 80 which can be solved by careful hardware design, 81 but there are also a large number of intricate system programs which must be written before one 82 has an adequate time-sharing system. An import-83 ant aspect of any future time-shared computer is 84 that until the system programming is completed, 85 especially the critical time-sharing supervisor, 86 the computer is completely worthless. Thus, it 87 is essential for future system design and imple-88 mentation that all aspects of time-sharing system problems be explored and understood in prototype 89 form on present computers so that major advances 90 in computer organization and usage can be made. 91

#### Acknowledgements

The authors wish to thank Bernard Galler, Robert Graham and Bruce Arden, of the University of Michigan, for making the MAD compiler available and for their advice with regard to its adaption into the present time-sharing system. The version of the Madtran Fortran-to-Mad editor program was generously supplied by Robert Rosin of the University of Michigan. Of the MIT Computation Center staff, Robert Creasy was of assistance in the evaluation of time-sharing performance, Lynda Korn is to be credited for her contributions to the pm and madtran commands, and Evelyn Dow for her work on the fap command.

## References

1. Strachey, C., "Time Sharing in Large Fast Computers," <u>Proceedings of the International</u> <u>Conference on Information Processing, UNESCO</u> (June, 1959), Paper B.2.19.

119

120

92

343

73

74



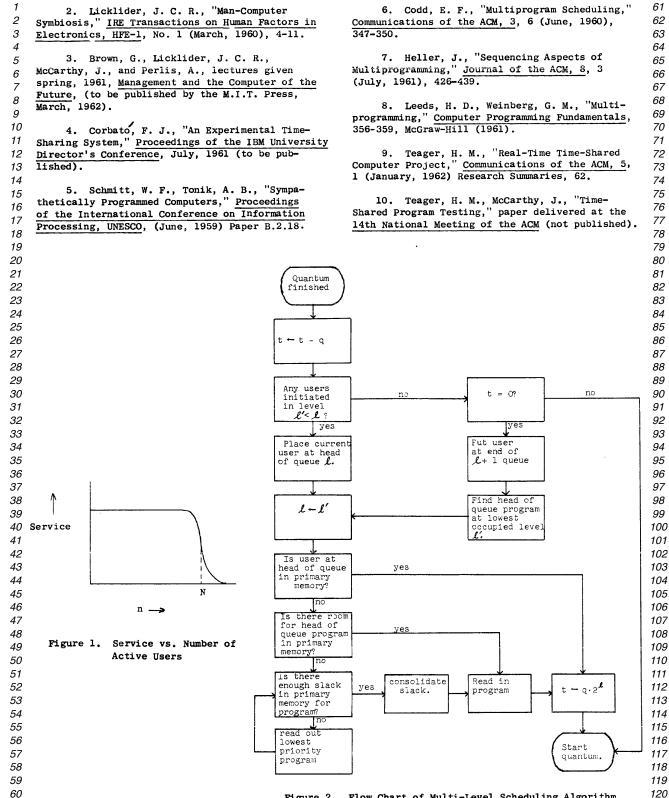


Figure 2. Flow Chart of Multi-Level Scheduling Algorithm