

Contents

1. Covering Multithreading Basics	1
Defining Multithreading Terms	1
Meeting Multithreading Standards	3
Benefiting From Multithreading	3
Improving Application Responsiveness.....	3
Using Multiprocessors Efficiently	3
Improving Program Structure	4
Using Fewer System Resources	4
Combining Threads and RPC.....	4
Understanding Basic Multithreading Concepts.....	5
Concurrency and Parallelism.....	5
Looking at Multithreading Structure	5
Scheduling	7
Cancellation	8
Synchronization	9

2. Basic Threads Programming	11
The Threads Library	11
Create a Default Thread	12
Wait for Thread Termination	14
A Simple Threads Example	15
Detaching a Thread	17
Create a Key for Thread-Specific Data	18
Delete the Thread-Specific Data Key	19
Set the Thread-Specific Data Key	20
Get the Thread-Specific Data Key	21
Get the Thread Identifier	25
Compare Thread IDs	26
Initializing Threads	27
Yield Thread Execution	28
Set the Thread Priority	29
Get the Thread Priority	30
Send a Signal to a Thread	31
Access the Signal Mask of the Calling Thread	32
Re-create and Reinitialize Critical Threads	33
Terminate a Thread	33
Finishing Up	34
Cancellation	34
Cancel a Thread	36
Enable or Disable Cancellation	37

Set Cancellation Type	38
Create a Cancellation Point	39
Push a Handler Onto the Stack	39
Pull a Handler Off the Stack.	40
3. Thread Create Attributes	43
Attributes	44
Initialize Attributes	45
Destroy Attributes	46
Set Detach State	47
Get Detach State.	49
Set Scope	50
Get Scope	52
Set Scheduling Policy	52
Get Scheduling Policy	54
Set Inherited Scheduling Policy.	55
Get Inherited Scheduling Policy	56
Set Scheduling Parameters	57
Get Scheduling Parameters	58
Set Stack Size	60
Get Stack Size	61
About Stacks.	61
Set Stack Address.	64
Get Stack Address	67
4. Programming With Synchronization Objects	69

Mutual Exclusion Lock Attributes	70
Initialize a Mutex Attribute Object	72
Destroy a Mutex Attribute Object	73
Set the Scope of a Mutex	74
Get the Scope of a Mutex	75
Using Mutual Exclusion Locks.	75
Initialize a Mutex	76
Lock a Mutex	78
Unlock a Mutex	79
Lock With a Nonblocking Mutex.	80
Destroy a Mutex.	81
Mutex Lock Code Examples.	82
Condition Variable Attributes	87
Initialize a Condition Variable Attribute	88
Remove a Condition Variable Attribute	89
Set the Scope of a Condition Variable	90
Get the Scope of a Condition Variable	91
Using Condition Variables	92
Initialize a Condition Variable	92
Block on a Condition Variable	94
Unblock a Specific Thread	96
Block Until a Specified Event.	98
Unblock All Threads	99
Destroy Condition Variable State.	101

The Lost Wake-Up Problem	102
The Producer/Consumer Problem	102
Semaphores	106
Counting Semaphores	107
Initialize a Semaphore	108
Named Semaphores	110
Increment a Semaphore	110
Block on a Semaphore Count	111
Decrement a Semaphore Count	112
Destroy the Semaphore State	113
The Producer/Consumer Problem, Using Semaphores . . .	114
Synchronization Across Process Boundaries	116
Producer/Consumer Problem Example	116
Interprocess Locking Without the Threads Library	118
Comparing Primitives	118
5. Programming With the Operating System	119
Process Creation–Forking Issues	119
The Fork-One Model	120
The Fork-All Model	124
Choosing the Right Fork	124
Process Creation–exec(2) and exit(2) Issues	124
Timers, Alarms, and Profiling	125
Per-LWP POSIX Timers	125
Per-Thread Alarms	126

Profiling	126
Nonlocal Goto—set jmp (3C) and long jmp (3C)	127
Resource Limits	127
LWPs and Scheduling Classes	127
Timeshare Scheduling	128
Realtime Scheduling	129
LWP Scheduling and Thread Binding	130
SIGWAITING—Creating LWPs for Waiting Threads	131
Aging LWPs	131
Extending Traditional Signals	132
Synchronous Signals	133
Asynchronous Signals	133
Continuation Semantics	134
Operations on Signals	135
Thread-Directed Signals	137
Completion Semantics	139
Signal Handlers and Async-Signal Safety	140
Interrupted Waits on Condition Variables (Solaris Threads, Only)	142
I/O Issues	144
I/O as a Remote Procedure Call	144
Tamed Asynchrony	144
Asynchronous I/O	145
Shared I/O and New I/O System Calls	146

Alternatives to <code>getc(3S)</code> and <code>putc(3S)</code>	147
6. Safe and Unsafe Interfaces	149
Thread Safety	149
MT Interface Safety Levels	151
Reentrant Functions for Unsafe Interfaces	152
Async-Signal-Safe Functions	153
MT Safety Levels for Libraries	153
Unsafe Libraries	154
7. Compiling and Debugging	155
Compiling a Multithreaded Application	155
Preparing for Compilation	155
Choosing Solaris or POSIX Semantics.	156
Including <code><thread.h></code> or <code><pthread.h></code>	156
Defining <code>_REENTRANT</code> or <code>_POSIX_C_SOURCE</code>	157
Linking With <code>libthread</code> or <code>libpthread</code>	157
Linking with <code>-lposix4</code> for POSIX Semaphores.	158
Link Old With New Carefully	159
Debugging Multithreaded Programs	159
Common Oversights	159
Tracing and Debugging With the TNF Utilities	160
Using <code>truss(1)</code>	161
Using <code>adb(1)</code>	161
Using <code>dbx</code>	161
8. Tools for Enhancing MT Programs	163

Scenario: Threading the Mandelbrot Program	164
Using Thread Analyzer to Evaluate Mandelbrot	165
Scenario: Checking a Program With LockLint	171
Scenario: Parallelizing Loops with LoopTool	176
For More Information	181
9. Programming with Solaris Threads	183
Comparing APIs for Solaris Threads and POSIX Threads	183
Major API Differences	184
Function Comparison Table	184
Unique Solaris Threads Functions	188
Suspend Thread Execution	188
Continue a Suspended Thread	189
Set Thread Concurrency Level	190
Get Thread Concurrency	191
Unique Solaris Synchronization Functions—Readers/Writer Locks	
192	
Initialize a Readers/Writer Lock	193
Acquire a Read Lock	195
Try to Acquire a Read Lock	195
Acquire a Write Lock	196
Try to Acquire a Write Lock	197
Unlock a Readers/Writer Lock	197
Destroy Readers/Writer Lock State	198
Similar Solaris Threads Functions	200

Create a Thread	200
Get the Minimal Stack Size.	203
Get the Thread Identifier	204
Yield Thread Execution.	204
Send a Signal to a Thread	205
Access the Signal Mask of the Calling Thread	205
Terminate a Thread	205
Wait for Thread Termination	206
Create a Thread-Specific Data Key	207
Set the Thread-Specific Data Key.	208
Get the Thread-Specific Data Key	208
Set the Thread Priority	208
Get the Thread Priority	209
Similar Synchronization Functions–Mutual Exclusion Locks .	210
Initialize a Mutex.	210
Destroy a Mutex.	211
Acquire a Mutex.	212
Release a Mutex	212
Try to Acquire a Mutex	212
Similar Synchronization Functions–Condition Variables	213
Initialize a Condition Variable	213
Destroy a Condition Variable	214
Wait for a Condition	215
Wait For an Absolute Time	215

Signal One Condition Variable.	216
Signal All Condition Variables.	216
Similar Synchronization Functions–Semaphores.	216
Initialize a Semaphore.	217
Increment a Semaphore.	218
Block on a Semaphore Count.	218
Decrement a Semaphore Count.	219
Destroy the Semaphore State.	219
Synchronization Across Process Boundaries.	220
Using LWPs Between Processes.	220
Producer/Consumer Problem Example.	221
Special Issues for fork() and Solaris Threads.	223
10. Programming Guidelines.	225
Rethinking Global Variables.	225
Providing for Static Local Variables.	227
Synchronizing Threads.	228
Single-Threaded Strategy.	228
Reentrance.	229
Avoiding Deadlock.	231
Deadlocks Related to Scheduling.	232
Locking Guidelines.	233
Following Some Basic Guidelines.	233
Creating and Using Threads.	234
Lightweight Processes.	235

Unbound Threads	236
Bound Threads	237
Thread Concurrency (Solaris Threads, Only).....	238
Efficiency.....	238
Thread Creation Guidelines	239
Working With Multiprocessors	239
The Underlying Architecture	240
Summary.....	245
Further Reading.....	245
A. Sample Application – Multithreaded grep	247
Description of tgrep	247
Getting Online Source Code.....	248
B. Solaris Threads Example: barrier.c	279
C. MT Safety Levels: Library Interfaces	285
Index.....	341

