ForkLight special program variables				
name	meaning	sharity	type	remark
`__P__`	total number of available threads	shared	`int`	read–only
`@`	current group ID	shared	`int`	redefinable by `fork`
`$`	current thread ID	private	`int`	automatically renumbered in `fork`

ForkLight function (pointer) type qualifiers			
name	meaning	remark	example for usage
`csync`	declare (pointer to) control-synchr. function		`csync int foo( int k ) {....}`
(none)	declare (pointer to) asynchronous function	default	`sh void (*task[Pmax])(void);`
ForkLight storage class qualifiers			
`sh`	declare a shared variable		`sh int k=1, *pk, a[32];`
(none)	declare a private variable	default	`int i,*pi, b[100];`

ForkLight language constructs: statements				
keyword	meaning	example for usage	region	body
`start`	all threads enter control-synchr. mode	`start { a[$]=$; }`	asynchr.	csync.
`join`	some threads enter control-synchr. mode	`join(;0;99) bustour();`	asynchr.	csync.
`retry`	jump back to the corresponding `join`	`else retry;`	asynchr.	—
`async`	enter asynchronous mode	`async puts("Hello");`	csync.	asynchr.
`seq`	one thread per group enters asyn. mode	`seq puts("Hello");`	csync.	asynchr.
`fork`	split current group in subgroups	`fork(3;@=$%3) fn(@);`	csync.	csync.
`===`	group–wide barrier synchronization	`f1(); ======= f2();`	both	—

ForkLight standard functions for atomic operations					
name	meaning	example for usage	mode	operands	result
`fetch_add(ps,ex)`	multiprefix sum	`$=fetch_add(&p,1);`	both	`int *`, `int`	`int`
`fetch_max(ps,ex)`	multiprefix maximum	`fetch_max(&m,a[$]);`	both	`int *`, `int`	`int`
`fetch_and(ps,ex)`	multiprefix bitwise AND	`k=fetch_and(&m,0);`	both	`int *`, `int`	`int`
`fetch_or(ps,ex)`	multiprefix bitwise OR	`k=fetch_or(&m,1);`	both	`int *`, `int`	`int`
`atomic_add(ps,ex)`	atomic increment	`atomic_add(&p,1);`	both	`int *`, `int`	none
`atomic_max(ps,ex)`	atomic maximization	`atomic_max(&p,1);`	both	`int *`, `int`	none
`atomic_and(ps,ex)`	atomic bitwise AND	`atomic_and(&p,1);`	both	`int *`, `int`	none
`atomic_or(ps,ex)`	atomic bitwise OR	`atomic_or(&p,1);`	both	`int *`, `int`	none

ForkLight memory allocation routines				
name	mode	type	parameters	meaning
`malloc`	async.	`char *`	`unsigned int`	allocate block on private heap
`free`	async.	`void`	`char *`	free block on private heap
`shmalloc`	async.	`char *`	`unsigned int`	allocate block on global shared heap
`shfree`	async.	`void`	`char *`	free block on global shared heap
`shalloc`	`csync`	`char *`	`unsigned int`	allocate block on automatic shared heap
`shavail`	async.	`unsigned int`	`void`	get size of free automatic shared heap space

ForkLight group structure inspection				
name	mode	type	parameters	meaning
`groupsize`	both	`int`	`void`	get the number of threads in my current group

Table 1: ForkLight at a glance.